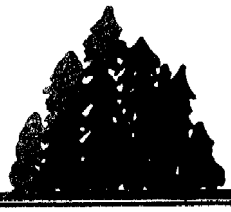
NORTHERN BYTES



Volume 5 Number 8

SEASON'S GREETINGS to all of our readers! These opening remarks are going to be relatively short, because it's 4 A.M. as I write this and because we've got a lot to go into this issue. But for starters, I'd like to point out that we often reprint articles from other newsletters. If we have the address of the author of a reprinted article, we will give him a free six issue subscription to NORTHERN BYTES (we also do this for articles contributed directly to this newsletter). We can't do this, however, if we don't have the author's address. So, if you see an article that you (or someone you know) wrote in Northern Bytes, please drop us a card and give us a current address, so we can update our mailing list.

We've had several folks ask us about subscriptions to Northern Bytes. Not yet. We still don't want to be obligated to produce a given number of future issues. As mentioned above, you can get a six-issue subscription absolutely FREE by submitting an article to NORTHERN BYTES (by the way, we occasionally slip up on this - we're only human. So, if you sent us an article, please check your mailing label to see if "issues remaining" count has been updated properly, and if it hasn't, please drop us a postcard!).

However, for those of you that simply want subscribe, we will make the following offer. If you have a VISA or Mastercard, you may send us your card number and expiration date, and we will automatically send you each new issue and bill your charge card for $2.00 (overseas airmail, $3.00) until either we stop publishing or you tell us not to do it anymore. Now some of you will object to this plan for one reason or another, and I agree that it leaves a lot to be desired. In fact, we're really not encouraging this kind of "subscription" (we're not actively promoting it, you'll notice), but we are offering it simply as a convenience for folks that hate to keep writing $2.00 checks for back issues that they've somehow missed. If you have a better idea, we'll be glad to consider it. If you do want to sign up, send your charge card number to NORTHERN BYTES in Sault Ste. Marie (not to The Alternate Source) because even though TAS will do the actual billing, your card number will be stored as part of our mailing list database, and that is maintained up here in the Great White North.

Speaking of our mailing list, we recently purged the clubs and user groups that have NOT been sending us exchange newsletters. Your group could be next, if you're not sending us your exchange newsletter. On the other hand, we appreciate all the exchange newsletters that we do get, and we often find a lot of really good information in them. So, if you belong to a TRS-80 user group, and if they publish anything resembling a newsletter, find out if they exchange newsletters with us.

Whatever happened to the TAS Public Domain Library? At present there are only four PD Library disks. The answer is that folks just haven't cooperated with us on this. We asked folks to send us one or two really good programs, preferably with some sort of documentation file. What we got were disks with notes saying "Here's the entire contents of the TRS-80 section on our local BBS. You sort through it and if you find anything good on it, go ahead and use it." How generous! I figure if it's not worth the sender's time to go through and cull out the obviously bad programs and maybe write up a short description of each good program, it's not worth my time, either. On top of that, we have found that a few folks have sent us "public domain" software that actually turned out to be copyrighted programs from Softside or CLOAD (with the copyright notices removed, of course). These lowlifes are the ones who have really killed our interest in the project - we don't want to be sued! Public Domain Library disk number 4 consists mostly of programs that have appeared in NORTHERN BYTES, and I suspect that trend will continue, since I usually have a pretty good idea of the origins of those programs. If you'd like to contribute a program to the Public Domain Library, PLEASE make sure you know the origin of it and PLEASE include some sort of descriptive text file on the disk.

I told you these remarks would be short. This is it - except for a gift suggestion for the season. If you've struggled with instruction manuals that just don't seem to make any sense at all (you say you haven't? You haven't been using computers very long, have you?), maybe you'd prefer one that's clearly written and easy to read for a change. Well, there is an instruction manual for all of us, and this year it's available in a new, easier to understand version called "The Book", available in bookstores everywhere. And, if you're celebrating Christmas, please remember that the true meaning of the day is not a new computer or peripheral. Unless you happen to be the one person out of 365.25 that was born on December 25, it's not YOUR birthday. It's HIS (or at least it's celebrated on that day). May all of our readers have a joyous Christmas and/or Hanukkah season, and best wishes to everyone for the new year!

THE EXTERMINATOR

Not too many BUGS to report this time, thankfully. In regard to Dave McGlumphy's "LINE" program (which appeared in NORTHERN BYTES Volume 5, Number 7, pages 4 & 5), the following comments were made by Donald G. Chapman, editor of the APCU (Association of Personal Computer Users) newsletter: "In line 170, DX=0 is never true as this was eliminated by the test IF ABS(DX)>ABS(DY) in line 130 that brings us to line 170. However, if DX and DY are both zero in the rare case of both coordinates being repeated by the RND function, then line 130 will branch to line 240 where DY=0 must be tested to avoid a divide by zero in DX/DY. Setting IX=1 in this case isn't mathematically right, but makes no difference as the loop executes only once with J=0."

And one more comment from an exchange newsletter. The Richmond TRS-80 Users Group newsletter (M.D. Kerby, editor) reports that Don Brate's program that patches TRSDOS 1.3 (which appeared in NORTHERN BYTES Volume 5, Number 6) had a bug in the patch that is supposed to increase the BASIC file loading speed by 50%. Apparently it does increase the loading speed, but in at least one case it duplicated the first five lines and added a line 12337. No fix for this one yet, so beware of it, and if you find the problem (or a fix for it!), please drop us a line!

TIME TO GO INSIDE!

"Well, it's time to go inside where it's warm. Yes the cool weather is coming and we will be staying inside more and hopefully using our computers...."

The above is reprinted from the opening remarks of the October, 1984 issue of the TBUG-80 newsletter. Now you might think that "T" stands for Tacoma or Toledo or Toronto or even Traverse City (or some other northern city). Wrong! TBUG (in this case) stands for Tampa Bay (TRS-80) Users Group (in case you flunked geography, Tampa Bay is in Florida). Wonder how those folks would describe one of our Northern Michigan winters (where it's not at all uncommon to have a few feet of snow on the ground, and temperatures of 30 degrees below zero)?

Which brings us to today's trivia question. TBUG-80 also has another meaning. Do you know what it is? Answer below!

* * * * * * * * * * *

Answer to trivia question: It was the first machine language monitor program for the TRS-80 Model I, and was sold by Radio Shack. If you actually bought and used this program, you are a true TRS-80 pioneer!

FOR SALE

IF YOU'RE STILL STUCK with a TRS-80 Level I here's your chance to upgrade to 48K Level II economically. -- For sale (A).- A TRS-80 CPU board with upper-lower case & twisted pair refresh modification. (B).- TRS-80 Expansion Interface with RS-232, Percom Double Density & Data Separator board, twisted pair refresh connections, gold plated connectors and CPU to EI buffer cable. All for $250. You connect your present key board to the CPU in your case and with your monitor - presto - 48K DD Level II. Write!- Al Hubbard, 5705 Junonia Spv., Fort Meyers, Florida 33908 1/813-466-4230.

Run your own non-commercial "unclassified" ad for only 10 cents per word per issue! Your name and address are FREE! Payment must be made in advance, VISA and MASTERCARD accepted.

LETTERS DEPARTMENT

A small sampling of the avalanche of mail received here at NORTHERN BYTES!

Date: Fri Oct 12, 1984 7:26 pm EDT **RECEIPT
From: David R. McGlumphy / MCI ID: 181-7759

TO: *Jack Decker / MCI ID: 102-7413

Good news! I found a calculator that does decimal/hexadecimal arithmetic and conversions for only $24.95 - at the Radio Shack computer center. It's a new item. It uses two AA batteries to drive an LCD display, and it can manipulate time figures so that you can calculate how many hours (or minutes or seconds) are between two times of the day. It also has a count-down timer from 99 minutes and 99 seconds as well as an alarm, a signal to beep every hour on the hour if you want, and a clock of course. Thank goodness someone came out with something to rival the TI-Programmer which costs about sixty bucks.

Bug report. The telephone dialer program I wrote in assembly will work better (in some cases, read that "will work") if you make the first instruction LD SP,41FEH to set the stack pointer. I use MODEM80/CMD for telecommunications, and by having the dialer's stack pointer set to 41FEH, I can use the "C" command of MODEM80 to dial the phone without having to reload MODEM80. Once I've dialed the number and have heard it begin to ring, I just press <BREAK> which normally exits to DOS from the dialer, but which instead returns to MODEM80. That MODEM80 sure is a smart piece of software.

Besides putzing around on my trusty model 1, I work with an IBM 370/148, (VSE release 3) though my boss may question the use of the word "work". IBM has to be concerned with security of all kinds because users have sensitive data. One of the tools IBM has provided is a command to ERASE a file when it is being deleted. (VSAM files, for those who wonder.) I haven't had to use that function, but I think it writes binary zeros to the disk file before the catalog (directory) entry is deleted. For no particular reason, I started wondering how such a function might be implemented on my machine.

I started thinking in terms of writing an assembly program because I would probably want to go from sector to sector through the target diskette. I thought of all the wonderful things I'd learned about the GAT sector which is the part of the directory that tells which grans have been used. That is, the Granule Allocation Table is changed whenever a program uses a gran of space somewhere on the diskette to write a file or whenever you save a program or file. The more I thought about having to read the GAT and deciphering its contents to determine which sectors to zero out, the more complicated my program-to-be became. I also thought about the multiple extents that show up in directory entries, but discarded that concern because the GAT information should be all I really needed.

I love NEWDOS80/V2. It has so many goodies in it. I can do so much with that operating system. Take DIRCHECK/CMD, for example. It has this neat function that allows you to clean up your directories. It doesn't fix farkled directories, but it does put binary zeros (as opposed to hex zeros?) in places where there used to be a directory entry but now has only remnant information because the file was deleted. Surely, thought I, such an operating system would have something in it that would do what I was considering. Lo and behold, it does! Almost. By using three DOS commands, I could accomplish most of what I'd thought about doing. And THAT, folks, is a lot faster than trying to write an assembly program.

DIR tells me how many free grans are available. I have five sectors per gran, so five times the number of grans tells me how many unused sectors there are on the diskette. Let's call that value X.

CREATE FILESPEC REC=X is the DOS command that creates a file with "X" records in it, and unless you tell it otherwise, the logical record length is 256, exactly the size of a sector. When DOS creates the file for you, he puts binary zeros in each of the records.

KILL FILESPEC needs no explanation.

The result? All of the initially unused sectors which could have held information like your love letters and which were still readable by someone using SUPERZAP or other utilities now contain binary zeros which may be less interesting than your old love letters.

Remember that I said these three steps would ALMOST cover up your tracks. If you save a BASIC program that uses only three sectors, the remaining two sectors in the gran remain unaffected. They'll still have in them what you left in them when you killed a file that used them. Is it worth figuring out a way to eliminate that left-over information? Not to me, it's not, but I had fun finding out what I've told you. Some people are into ADVENTURE games. This is my kind of adventure.

By the way, an alternative to using the three DOS commands is to format a diskette and then copy-by-file...not much fun if you have only one drive.

One more tidbit. When I format a diskette, I like to have it identified as mine, in case I send it out, and I like to have the diskette-ID show up in lowercase letters. If you look in the NEWDOS80/V2 manual in section 5.2 (DOS System Modules), you'll find that the format command is in file SYS6/SYS. Using SUPERZAP's DFS (Display File Sectors) command to look at SYS6/SYS, you'll find the default diskette name in FRS (File Relative Sector) 0CH (12D) at address 3DH. As it comes from the factory, the default is "NOTNAMED". I use SUPERZAP to put in most of my last name in lowercase. The limit is eight characters, so I couldn't put in "Humble Dave, C.D.P.".

A final note. I have an elapsed time indicator on my Model I. Right now it says we've used this infernal machine for three thousand, eight hundred, ninety-six hours. I also have such an indicator in my '75 Dodge Dart, and I found that the car averages almost exactly thirty-five miles per hour. I generally drive right at the speed limit. Look at it this way - at 35 MPH fo 3,896 hours, I'd have driven my car one hundred thirty-six thousand, three hundred sixty miles in the same amount of time we've driven my Model I. The difference is that notwithstanding the time I modified the keyboard with a glass of lemonade, the computer hasn't required repairs. Let's see now. On a dollar per hour basis, just how expensive IS my hobby?

So tell me, Ernestine, what have you learned about your 'puter lately?

-Dave McGlumphy MCI# 181-7759.

Date: Sun Oct 14, 1984 10:07 am EDT **RECEIPT
From: David H. Bower / MCI ID: 224-9092

TO: *Jack Decker / MCI ID: 102-7413
Subject: 4P

Jack,

I've got a problem you're readers might be able to help me with.

I've come across a couple of the new Model 4P's (they're the new ones with the green screen, but I don't know if that matters) that seem to have bad drives. They won't pass Floppy Doctor and CRC errors seem to be a real problem. Backups become impossible and even booting is a problem. I've aligned the two drives in one of the 4P's and they are perfect. The green screen Model 4P has a new CPU board and I'm wondering (afraid) if they might have messed up the drive capabilities like they did on the Model III shield modification.

I've also run into some problems with Model 4's and 80 track drives. A customer of ours has upgraded all his 4's with two 80 track drives per computer and has had nothing but trouble.

I'd like to hear about any problems anybody has had with the 4 and 4P, and any fixes!

The green screen 4P has a new CPU board and the jumper for the extra 64K of ram has been changed. It is now E1, E2 and E3 I believe and apparently it's not quite in the same place. I can find out for sure if someone out there can't verify it.

Also there are TWO versions of the green screen Model 4P. The difference is in the keyboard storage compartment. One has a piece of styrofoam along the back to stop the keyboard and the keyboard is kind of loose.

The other has 3 L-shaped (upside down) rubber bumpers that hold the keyboard very snug. But the back bumper hangs down. But if you force the keyboard in without making sure the back bumper is up you could eventually (within a few openings & closings) mess up some wiring back there. The keyboard should slide in with NO force.

A friend had the keyboard problem. Sometimes it worked, sometimes it was dead. When he took it back to the computer center it was giving hard disk error messages, even though there was no hard disk. The Radio Shack salesman at the Radio Shack Computer Center said that the hard disk messages were advertisements put in the ROM by Tandy to try and get us to buy a hard disk. The sad part is that the Jerk was serious and didn't want to take the computer in for repair. If I sound a little abusive it's because I had a run in with the Jerk myself. He is by far the most ignorant Radio Shack salesman I have ever talked to.

One more thing. You said that removing the carriage returns from a transmitted file was a problem. I suggested a Soft Key in Allwrite to do it. But you don't need to remove them. Allwrite ignores the carriage returns at the end of the lines, unless you have two in a row. So they will only affect your video formatting.

Dave

Date: Wed Nov 07, 1984 8:31 pm EST **RECEIPT
From: International Test 6 / MCI ID: 248-6778

TO: *Jack Decker / MCI ID: 102-7413
Subject: Newdos/80's Parameter Scanner Corrections

Jack:

Since you published my article on NEWDOS/80's PARAMETER SCANNER in Northern Bytes Volume 5, Number 4 I have made one further discovery about the routine.

My commented disassembly for the Model I version shows the following at 4CEAH:

```
4CEA    3E34    LD      A,34H    ;Set A to 34H
```

The comment is downright cryptic -- but I simply wasn't sure what the author was doing. Now I know.

Storing 34H in the A register allows a CALL or JP to one of Newdos/80's error handling routines which will result in:

ILLEGAL KEYWORD OR SEPARATOR OR TERMINATOR

So this allows a programmer to use the Newdos/80 error message if the condition flags are NOT as expected.

Greg Small, Box 607, Stouffville, Ontario L0H 1L0, CANADA

[Note to readers: In case you are wondering about the strange message header, it's because Greg is one of the folks helping MCI Mail to test their foreign access service, scheduled to be available to the general public on or about January 1, 1985. It appears that Canadians will be using Bell Datapac to access the Telenet network (as is presently done to access CompuServe), then accessing MCI Mail through Telenet, but this is NOT official and could change. However, for a service that is not "officially" available yet, a lot of Canadian organizations seem to have MCI Mail ID numbers already! A quick check using MCI Mail's CREATE command turned up the following:

CANADA:
212-9220 Agricultrual Canada, 930 Carling Avenue, Ottawa, ON
113-5150 Apple Canada, Apple Canada Don Mills, ON
112-5201 Bell Canada, Toronto, ON
206-4721 Environment Canada, 506 W. Burnsi, Victoria, BC
217-8954 Environment Canada, Departmental Ottawa, CA
231-1825 Statistics Canada, Ottawa, Canada
230-1840 Transport Canada, Ottawa, Canada
217-9092 Warner-Lambert Canada Ltd, Scarborough, OT

ALBERTA:
232-1732 University of Alberta, Cameron Library, Alberta, Canada
239-1849 University Of Alberta, Alberta, Canada

BC:
239-1852 University of B.C., U of Brit Columbia, Vancouver B.C.

ONTARIO:
233-1843 TV Ontario, Toronto, Canada
212-9055 University Of Western Ont, 1137 Western, Ontario, CA

The really interesting entries on the above list are Apple Canada and Bell Canada, because from the low-numbered prefixes, it would appear that they have had accounts for quite some time (the prefixes start with 100, and although not issued in fully chronological order, I.D.s with the same or similar prefixes were issued around the same time). This strikes me as a bit odd, because all summer MCI Mail has claimed that service was not available in Canada. Of course, just about all of the above organizations are large enough to have U.S. "foreign exchange" telephone lines through which they could access a U.S. MCI Mail phone line (or maybe they just dialed direct into the U.S.?), so maybe it's not so strange.

Anyway, if you live outside the U.S.A. and would like to get onto MCI Mail, here's the person to contact: Mr. William Byrne, MCI Mail, 2000 M. Street N.W., Suite 300, Washington, D.C. 20036 (or by telephone at (202) 463-3281 or telex 6502194251, answerback 6502194251 MCI). And, if you want to send an MCI Mail letter to Greg Small, I would suggest trying the user name GSMALL before you use the ID number given above, since that number is most likely temporary and Greg will probably get his own number when the test period is over.]

Dear Jack,

At school we have a network of 15 model 4 TRS-80's. It is the Network 3 version 2.1.4. The 14 terminals have a boot ROM which initializes the network from the double disk host machine thru the RS232 port. The default baud rate and the boot-up baud rate is 9600. This sounds fast until all 14 try to get on line at once. CLOADing them all at once at 500 baud is faster than the sequential loading from the disk. The system gets over loaded, hangs up, and ignores those who are trying to boot up, favoring those who have booted first.

It has other peculiarities, such as not being able to read a directory from BASIC, having to re-load BASIC and program if you go to the system level to read the directory. Not even Model I TRSDOS was that bad.

It does save some money as we only have to have the one disk machine, and only one printer which can be used by any of the terminals. It is however a dumb terminal, displaying only the number of the accessing terminal, and sometimes displays the wrong number or an impossible one. It has to be on line to service the terminals and cannot be used for programming, etc.

Although the initializing baud rate and the default rate is 9600, after a terminal comes on line it can request 19,200 baud. When all terminals are up, the host can be reset to run 19,200 also. This works very well, until a terminal hangs or is reset and must be re-initialized. At that time to bring it back on line, you must then reset the host to 9600, reload the terminal, and then reset the host back to the high rate, which is a hassle. Even though I have spent HOURS on the 800 educational number, plus talking to the writers of the network program (DOSPLUS), I have gotten nowhere. If it was possible for the host to recognize the incoming baud rate and transmit at that rate, such as some to the smart modems it would make the network run 100% better. I would appreciate your advise or help in this matter. Maybe you can include the problem in Northern Bytes.

Thanks again for your past help. I really appreciate your publication and the public domain disks.

Sincerely, Cy Galley 3318-26th Avenue, Rock Island, Illinois 61201

Dear Jack,

I was just making up an index of items of interest to me in Volume 5, Number 4 and re-read the "Model III Disk Basic Bug?" item at the bottom of page 20.

I discovered the same "feature" several years ago and "solved" it as follows:

```
L=20: FIELD 1, L*1 AS A$
```

Yes-merely add the "*1" after the (numeric) variable and it works. WHY? you ask? I DON'T KNOW BUT IT DO, I reply.

At the same time I came across this, I uncovered the "fact" that you should NOT remove spaces (i.e. compress) FIELD statements. The "compress" utility of TRSDOS 1.3 does not, and if you do it, you'll get a "SYNTAX ERROR". Another "IDKBID".

- Nate Salsbury

[The following is a portion of a letter sent to The Alternate Source. They felt that perhaps Northern Bytes readers might like a crack at this one.]

My question concerns the Radio Shack TRS-80 Model 4 computer running in Model 4 mode under TRSDOS 6. The specific problem is the interfacing of Basic to Z-80 assembler and thereby the access of the DOS service routines via the two way passing of information between Basic and assembler

I have been trying to access service routine #15,1 using CALL M%(RP%,CP%,BP%) where:
M% = address of the object code
RP%= pointer to R% which holds the screen row to read
CP%= pointer to C% which holds the screen column to read
BP%= pointer to B% which should hold the byte located at R%,C% on the screen after returning from the assembler routine

I have not been able to successfully pass data to and from the assembler routine. I have been able to access service routine #15,6 successfully but that requires no data passing. Since the Radio Shack documentation is not at all clear on this subject and since there does not seem to be any Model 4 documentation on the market which clearly describes this type of interface, the public would be well served by your newsletter publishing a complete example (including object code) of the Basic/assembler interface. This example should show a Basic routine which uses a three-variable CALL statement to access an assembler routine which then uses the passed data and passes information back to the Basic routine. The assembler routine should access at least one DOS service routine.

- Peter G. Raeth, 5608-04 Bismach Drive, Alexandria, Virginia 22312

[Readers, if anyone would like to try to help Mr. Raeth, you may contact him directly, or you may send us an article of the type he has described and we'll be happy to publish it here in Northern Bytes!]

OVERSEAS EXPERIENCE
by "Computer Nut"

Introduction

There can be no doubt about it. I am a computer nut. Computers and programming have been a way of life for me for about 18 years.

Originally computing was restricted to normal working hours and normal overtime hours (usually), and the remainder of the time was spent reading computer books and magazines. For over five years, since the arrival of the TRS-80 Model I, computers have filled both my working hours and my leisure hours.

My computer experience has been gained overseas, relative to the United States, on two almost diametrically opposite points on the globe. It is my intention to relate some of my experiences in the microcomputer world, and to comment on some of the practices and trends as I see them.

Hardware

At the time when I was ready to make my first computer purchase, there were three contenders in the marketplace:- Apple, Pet, and TRS-80. Several other possibilities were eliminated since I had very little hardware experience, and the thought of assembling a computer was not my idea of happiness. There were private importers for each of the systems in the country where I live, but the only one I could try before buying was the Pet.

Eventually the choice fell on the TRS-80, mainly because of the expansion possibilities and the price.

All three systems were manufactured in the United States, where electricity is supplied in alternating current at about 110 volts and 60 cycles per second. In many (if not most) other countries the voltage is somewhere between 210 and 240, at 50 cycles per second.

One thing you do NOT do successfully is connect a 110 volt computer to a 220 volt power supply. You have two main options. One is to convert the computer to 220 volt operation, the other is to provide an external step-down transformer for producing a 110 volt supply.

The TRS-80 was run off an external power pack, and Tandy had thoughtfully provided a special power pack for 220 volt operation. The screen was also an original Tandy, but it was converted internally for 220 volt operation, using a step down transformer built into the screen case.

The screen conversion was an object lesson in how NOT to do it. The step down transformer was wound on a rectangular iron core, and placed within about 2 inches of the tube. If you want to see the effect a magnet has on a CRT tube, try holding a small magnet near the front of a screen containing text or other information. Now try shaking it backwards and forwards 50 times each second, and you will see the effect of having a transformer placed next to the tube.

The solution is to move the transformer further away. In this case it was moved outside the screen cabinet.

The effect of the 50 Hz instead of the 60 Hz is more difficult to determine. There are two possible problems. The first is that less iron core is needed in a 60 Hz transformer than in a 50 Hz transformer at the same voltage. So if the power supply transformer is dimensioned to the exact requirements at 60 Hz it may overheat and melt if used at 50 Hz.

This problem was experienced by a friend of mine who bought one of the very popular dot matrix printers in the United States, and tried to run it temporarily at 110 volt, 50 Hz. The printer operated satisfactorily, but not for very long.

The other potential problem is that the hardware may in some way depend on the mains frequency. A common example is the synchronous motor in 8 inch floppy disk drives. Some drives were provided with two different sets of pulleys, one for 50 Hz operation and one set for 60 Hz.

Another common example is the TRS-80 Model III which uses the mains frequency for the real-time clock and the screen update. Fortunately Tandy has provided solutions for these situations, a hardware jumper for the video section, and a software patch for the real-time clock.

Many power supplies produced today have jumpers for 110 volt or 220 volt operation, and it is seldom a problem for manufacturers to avoid using the mains frequency for control purposes. The suggestion to manufacturers, therefore, must be to use power supplies which can easily be converted to all the common mains voltages, and to avoid relying on the mains frequency for control.

A hardware related problem which is independent of power supplies, and which does not affect integrated computers such as the Model 4, is the proliferation of TV color systems. The United States, Canada, Japan, and a few other countries, use the NTSC color system (often quoted as being an acronym for Never The Same Color, but actually National Television Standards Committee). Most European, South American, and Australasian countries use the PAL (Phase Alternation Line) color system, with France and some Eastern European and African countries using the SECAM color system (SEquentiel Couleur A Memoire).

Color computers with TV modulators produced in the United States obviously use the NTSC color system and these cannot directly be used with most TV sets used in countries with PAL or SECAM color systems. Multi-standard TV sets are available, usually at extra cost, which can decode more than one color system.

The screen refresh rate is usually related to the mains frequency, with 60 Hz giving 60 screen updates each second, and 50 Hz giving 50 screen updates each second. The screen is divided into a number of horizontal lines, usually 525 lines for a 60 Hz refresh rate, and 625 lines for a 50 Hz refresh rate. A computer generating a 525 line display can often be used with a TV set up for 50 Hz operation, but the picture may be slightly unstable and will probably not cover the entire screen.

Apart from color systems and refresh rate a TV channel is also intended to reproduce sound. A computer system using the TV monitor for sound reproduction can expect even more problems when being used in another country. Unfortunately there is no universally agreed standard for incorporating the sound in a TV channel. Some use AM (amplitude modulated) encoding, and some use FM (frequency modulated) encoding. Some have the sound carrier below the video carrier, others have it above, and at various distances from each other.

If you don't understand all this, don't be unduly worried. The main thing to understand is that there are problems if:

1. A TV set is used for color reproduction from a modulated computer signal, and
2. A TV set is used, with or without color, at a different screen refresh rate, and
3. A TV set is used for sound reproduction from a computer generated signal.

After some time, especially if the manufacturer of a computer system wishes to sell the product outside North America, a modification will usually be made available for converting a computer from one color system to another. However the cost of conversion may be excessive, and a better solution may be the purchase of another computer.

From the letters I have received from users who wish to make a move from the U.S. to an overseas country, the information provided by the computer manufacturers in the U.S. is usually far from helpful. I would suggest contacting the importer or subsidiary at or near the intended destination. They will often have experience in making any required conversions.

Tandy usually provides three subsidiary addresses outside North America. They are:

Parc Industriel De Naninne
5140 Naninne
Belgium

Bilston Road
Wednesbury
West Midlands WS10 7JN
United Kingdom

and

91 Kurrajong Road
Mount Druitt
NSW 2770
Australia.

That's enough about hardware for now. Next time we shall look at some hardware/software problems.

-"Computer nut"

PostScript: For those who call me an expert, my definition of an expert is someone who is a trained idiot in his own field, and a common idiot in all other fields.

MACRO-80 PATCH

I don't know the origin of these patches, but they are supposed to make MACRO-80 from Microsoft (M80/CMD) run on the Model III. The first three change the HIMEM pointer from 4049H (Model I) to 4411H (Model III), and they go like this: In M80/CMD, starting at addresses 9235H and 9ED3H, and in L80/CMD starting at address 5213H, change the three bytes 2A 49 40 to 2A 11 44 (that's three changes of three bytes each).

Also, on L80/CMD, the DOS call "FEXT" (inserts a default name extension into a filespec) must be changed for anything other than NEWDOS/80 and on the MAX80. For (Model I) TRSDOS 2.3 and on both the Model I and III versions of NEWDOS/80, the address is 4473H, while for other Model III DOSes and on the MAX80 the corresponding address is 444BH.

DR - a disk route program for NEWDOS/80 version 2

by Arne Ronde

1. Introduction

Newdos/80 has a ROUTE command which allows display or printer output to be routed to the display, printer, or null file or any combination of these. On the Model III the RS-232 output can also be rerouted. The keyboard input and, on the model III, RS-232 input, can also be rerouted.

The route command allows the possibility of rerouting to a user-defined routine in memory, with the MM=address parameter, but no indication is given in the manual how this routine can be written. The spool system and the DO or CHAIN command allows for rerouting printer output and keyboard input to a disk file, but with some restrictions.

The description of the spooler states that the system overlay files do not expect output to be rerouted to a disk file, so the spooler attempts to ignore output from the SYS files. There is no reason given for the restriction on system file output. Routing of input and output can only be used with programs which use the device control blocks provided by the system. Display output which is poked directly into the display memory will not be caught by a reroute routine, and similarly for printer and RS-232 output.

2. Rerouting to disk

Any attempt to write a routine which will reroute output to a disk file soon gives an indication that the warning about system output is correct. The system can easily hang or reboot or destroy the disk contents. This occurs when the disk output file is to be extended, indicating that during file extension a system file is loaded, thus destroying the system file which was producing the output being rerouted.

If system output is to be rerouted to disk, a necessary condition is that loading of system overlays can be avoided. An alternative possibility, used by the print spooler, is to ignore output from the system files.

An attempt to suppress system output gives the result that there is no simple and unique method for determining when the output comes from a system file. The method used by the spooler is to test for a flag which is set whenever a system file is loaded, and reset when it terminates. Unfortunately, at least one of the system files clears this flag while still actively sending output to the display, or printer and display if the display is rerouted, and the spooler will not catch this condition.

The alternative, to eliminate all system overlay loading, can, in fact, be implemented by following a number of guidelines. The most important is that the file must not be extended during system output. This means in effect that the output file must be created with its maximum length before rerouting takes place. It also means that the routine must determine the length of the file and stop output as soon as the end of the allocated space is reached.

Another problem surfaces when system overlays during rerouting are avoided. Some of the system files assume that they have direct access to the disk drives, and will not reselect a drive for each disk operation. This if a directory listing for drive 1 is routed to a file on drive 0, then as soon as drive 0 is required for disk output, the remainder of the directory will be displayed from drive 0. The solution is to reselect the drive which was selected before disk output took place from the disk routing program.

The enclosed program, DR/CMD, allows for routing all display or printer output to a disk file. If the output file is allocated before the program is loaded, then only the assigned file size will be used. Otherwise the program will attempt to allocate a file containing 512 sectors. This (128K) should be sufficient for most needs, but if larger files are required they can be preallocated using the CREATE command.

The file is filled with binary zeroes, and the number of sectors in the file is counted. This count is stored and used during the output phase. Once the count reaches zero, further output is ignored. The DR program must be called again to close the output file and to deroute the output.

For each character to be sent to a disk file, the currently selected drive number is stored, and if this changes during disk output, then the previous drive is reselected. This may not actually be necessary, and may give excess disk activity, but it ensures that system output can be sent to a disk file. The reselect code could be avoided if system file output was ignored.

3. Program usage

The command used to reroute to disk is similar to the ROUTE command, except that a filename is specified instead of an address in memory. The program replaces this filename with the actual address of the output routine, and passes on the reconstructed command to DOS as a ROUTE command.

To route the display output to the display and to a disk file called DISPLAY/RTE on drive 1, the following command could be used:

DR DO,DO,MM=DISPLAY/RTE:1

where DO as the first parameter indicates display output is rerouted. DO as the second parameter indicates that the output is sent to the display, and the third parameter indicates that it will also be sent to a disk file. The program will rebuild the command to read:

ROUTE DO,DO,MM=xxxxH

where xxxxH is the starting address in memory of the rerouting code, and send the command to the display and to DOS.

The program will relocate to high memory, and will modify the high memory value to protect the code. If programs do not respect high memory, then they should NOT be used with the reroute routine.

The parameters which can be specified for rerouting are DO for display output, PR for printer output, KB for keyboard input, and, on the Model III only, RI for RS-232 input and RO for RS-232 output. The filename to be used must follow the MM= parameter.

To deallocate a rerouting, the command format is similar. If only the first parameter is specified, the output rerouting will be terminated completely. If other parameters are specified, then the previous routing will be cleared, and the new specifications will be put into effect. Any disk file which is open for rerouting will be closed before the new file is allocated and opened.

If the routine being removed from memory is placed just above the high memory pointer, then high memory will be returned to the system. If possible, disk routes should be deallocated in the opposite sequence in which they are allocated. For example:

DR,DO,DO,MM=DISPLAY/RTE
DR,PR,PR,MM=PRINTER/RTE

execute programs producing output, followed by

DR,PR
DR,DO

This first deallocates the printer route, which was placed at the lowest memory address, and resets the high memory pointer to point to the display routing routine. The next command then deallocates and closes the display output route, and restores high memory to its original value. If the deallocations had been performed in reverse sequence, DO followed by PR, then only the space used by the display route would be restored to the system.

It should be noted that the ROUTE command can be us instead of the DR program to deallocate the route in effect, but t file will not be closed in this case. An output file will therefore not have the end of file pointer set correctly. To avoid mistakes the DR command can be used in all cases where the route command would be used, except when the route command is called directly from a program.

The routing routines require 256 bytes for the disk sector buffer, 32 bytes for a file control block, 16 bytes for the system pointers, and about 49 bytes for the input or output code, giving about 353 bytes altogether. The same amount is used for both input and output routes.

4. Conclusion

Rerouting of system input and output can be performed as long as some precautionary measures are taken, and some restrictions implemented. It can be useful for providing a record of information which has been sent to the screen, for later analysis, or for repeated printing, or other processing. Sending directory listings to a disk file could, for example, be used for a disk indexing program which could read the rerouted output. Printer output could be routed to a file for later printing, or for printing on another computer or printer.

If a little care is taken with allocating disk file space then the routine should be useful for many purposes. It will ONLY operate with NEWDOS/80 version 2. Any attempt to use it on other systems will give unpredictable results. The program could be expanded to provide for input or output to media other than disk. An example would be to provide RS-232 input and output on the Model I.

```
                00100          ;TITLE  '<DR/ASM version 84.07.19>'
5200            00110          ORG     5200H
5200            00120 BEGIN    EQU     $
4033            00130 ROUTAS   EQU     4033H           ;route address start
4420            00140 OPNOPT   EQU     4420H           ;open file
4424            00150 OPNIPT   EQU     4424H           ;open input
4428            00160 CLSFIL   EQU     4428H           ;close file
4049            00170 HIMEM1   EQU     4049H           ;himem mod 1
4411            00180 HIMEM3   EQU     4411H           ;himem mod 3
4405            00190 EXDOSC   EQU     4405H           ;execute dos command
4409            00200 DSERRH   EQU     4409H           ;error exit
```

```
4369            00210 OVLFG1  EQU   4369H          ;overlay active dos
4289            00220 OVLFG3  EQU   4289H          ;ovlay active mod 3
4419            00230 DOSCMD  EQU   4419H          ;dos command
4439            00240 WRTSEC  EQU   4439H          ;write disk sector
445B            00250 SELDRV  EQU   445BH          ;select drive
4308            00260 PRVDRV  EQU   4308H          ;prev drive mod 1
427E            00270 PRVDR3  EQU   427EH          ;prev drive mod 3
444B            00280 ALLOCF  EQU   444BH          ;allocate space
4412            00290 POSFCB  EQU   4412H          ;position file
443F            00300 POSBEG  EQU   443FH          ;position file begin
4467            00310 DISPMS  EQU   4467H          ;display message
5200 31BA55     00320         LD    SP,STAK        ;stack pointer
5203 E5         00330         PUSH  HL             ;cond addr
5204 216F54     00340         LD    HL,RTEBUF      ;route command buffer
5207 223754     00350         LD    (NBUFOT),HL    ;next output byte
520A 115654     00360         LD    DE,TTBL1-3     ;for mod 1
520D 3A5400     00370         LD    A,(54H)        ;ind mod 3
5210 3D         00380         DEC   A
5211 281F       00390         JR    Z,INMOD1
5213 114C54     00400         LD    DE,TTBL3-3     ;for mod 3
5216 211144     00410         LD    HL,HIMEM3      ;himem addr
5219 228E52     00420         LD    (HMA1),HL
521C 229E52     00430         LD    (HMA2),HL
521F 22C952     00440         LD    (HMA3),HL
5222 221D53     00450         LD    (HMA4),HL
5225 221653     00460         LD    (HMA5),HL
5228 23         00470         INC   HL             ;next byte
5229 229452     00480         LD    (HMAP),HL
522C 217E42     00490         LD    HL,PRVDR3      ;prev drive #
522F 22EE53     00500         LD    (PDRA),HL      ;save addr
5232            00510 INMOD1  EQU   $
5232 E1         00520         POP   HL             ;cond input
5233 223454     00530         LD    (NBUFIN),HL    ;pointer to next move
5236            00540 FNTYEN  EQU   $              ;find type entry
5236 13         00550         INC   DE
5237 13         00560         INC   DE
5238 13         00570         INC   DE             ;to entry name
5239 1A         00580         LD    A,(DE)         ;first char
523A B7         00590         OR    A
523B CAA853     00600         JP    Z,SENRBY       ;not found, send cond
523E BE         00610         CP    (HL)           ;check with cond line
523F 13         00620         INC   DE
5240 1A         00630         LD    A,(DE)         ;next char
5241 13         00640         INC   DE
5242 20F2       00650         JR    NZ,FNTYEN      ;not found
5244 23         00660         INC   HL             ;else next char
5245 BE         00670         CP    (HL)
5246 2B         00680         DEC   HL
5247 20ED       00690         JR    NZ,FNTYEN      ;not this entry
5249 23         00700         INC   HL
524A 23         00710         INC   HL             ;to next char
524B CD1D54     00720         CALL  CHCHTY         ;check type
524E DAA853     00730         JP    C,SENRBY       ;type not found
                00740 ;clear route information for type found
5251 E5         00750         PUSH  HL
5252 D5         00760         PUSH  DE
5253 13         00770         INC   DE             ;to addr
5254 EB         00780         EX    DE,HL
5255 5E         00790         LD    E,(HL)         ;orig fcb addr
5256 23         00800         INC   HL
5257 56         00810         LD    D,(HL)
5258 2A3440     00820         LD    HL,(ROUTAS+1)  ;route pointer
525B            00830 CHNXRT  EQU   $
525B 23         00840         INC   HL             ;to ptr next fcb
525C 7E         00850         LD    A,(HL)
525D 23         00860         INC   HL
525E 66         00870         LD    H,(HL)         ;ptr msb
525F 6F         00880         LD    L,A            ;ptr lsb
5260 B4         00890         OR    H
5261 2840       00900         JR    Z,CLROUX       ;end of routes, exit
5263 7E         00910         LD    A,(HL)
5264 BB         00920         CP    E              ;check required
5265 23         00930         INC   HL
5266 7E         00940         LD    A,(HL)
5267 23         00950         INC   HL
5268 20F1       00960         JR    NZ,CHNXRT      ;not reqd fcb
526A BA         00970         CP    D              ;fcb msb
526B 20EE       00980         JR    NZ,CHNXRT      ;not found
526D 7C         00990         LD    A,H            ;check in dos area
526E FE52       01000         CP    52H
5270 38E9       01010         JR    C,CHNXRT       ;yes, leave entry
5272 F3         01020         DI                   ;no interrupt in delete
5273 E5         01030         PUSH  HL             ;save addr
5274 010600     01040         LD    BC,6           ;to fcb part
5277 09         01050         ADD   HL,BC
5278 36C9       01060         LD    (HL),0C9H      ;set to ret
527A 44         01070         LD    B,H            ;save value
527B 4D         01080         LD    C,L
527C 23         01090         INC   HL             ;to addr ptr
527D 71         01100         LD    (HL),C
527E 23         01110         INC   HL
527F 70         01120         LD    (HL),B         ;pointer to code
5280 013200     01130         LD    BC,OTL+6       ;offset for fcb
5283 09         01140         ADD   HL,BC
5284 EB         01150         EX    DE,HL          ;fcb addr to de
5285 CD2844     01160         CALL  CLSFIL         ;close file
5288 EB         01170         EX    DE,HL          ;fcb addr to hl
5289 01C3FF     01180         LD    BC,-OTL-16-1   ;addr of himem
528C 09         01190         ADD   HL,BC
528D 3A4940     01200         LD    A,(HIMEM1)     ;check if lowest
528E            01210 HMA1    EQU   $-2
5290 BD         01220         CP    L
5291 2000       01230         JR    NZ,LVHIME      ;no, leave himem
5293 3A4A40     01240         LD    A,(HIMEM1+1)
5294            01250 HMAP    EQU   $-2
5296 BC         01260         CP    H              ;check if msb
5297 2007       01270         JR    NZ,LVHIME
5299 015C01     01280         LD    BC,256+32+OTL+16 ;len used
529C 09         01290         ADD   HL,BC
529D 224940     01300         LD    (HIMEM1),HL    ;save new
529E            01310 HMA2    EQU   $-2
52A0            01320 LVHIME  EQU   $
52A0 E1         01330         POP   HL             ;restore ptr
52A1 1808       01340         JR    CHNXRT         ;check next route
52A3            01350 CLROUX  EQU   $
52A3 D1         01360         POP   DE             ;restore adrs
52A4 E1         01370         POP   HL
52A5            01380 RPNEXT  EQU   $              ;find next param
52A5 06FF       01390         LD    B,0FFH         ;MM count
52A7            01400 INNEXM  EQU   $
52A7 04         01410         INC   B              ;incr MM count
52A8            01420 FNEQSN  EQU   $
52A8 CD1D54     01430         CALL  CHCHTY         ;check type
52AB CAB153     01440         JP    Z,SENROU       ;end of mesg
52AE 30F5       01450         JR    NC,RPNEXT      ;delimiter, clear count
52B0 23         01460         INC   HL
52B1 FE4D       01470         CP    'M'            ;check MM
52B3 28F2       01480         JR    Z,INNEXM       ;incr M count
52B5 FE3D       01490         CP    '='            ;check MM=
52B7 20EF       01500         JR    NZ,FNEQSN      ;not found
52B9 78         01510         LD    A,B            ;get M count
52BA FE02       01520         CP    2
52BC 20E7       01530         JR    NZ,RPNEXT      ;not 2, try next
52BE 7E         01540         LD    A,(HL)         ;next char
52BF FE41       01550         CP    'A'            ;check if alpha
52C1 38E2       01560         JR    C,RPNEXT       ;no, pass thru
52C3 CD3154     01570         CALL  XFCMLN         ;xfer to cond line
52C6 D5         01580         PUSH  DE
52C7 E5         01590         PUSH  HL             ;save adrs
52C8 2A4940     01600         LD    HL,(HIMEM1)    ;current high
52C9            01610 HMA3    EQU   $-2
52CB 25         01620         DEC   H              ;sub 256
52CC 23         01630         INC   HL             ;for allowed start
52CD 223A53     01640         LD    (BUFRAD),HL    ;buffer addr
52D0 1A         01650         LD    A,(DE)         ;check output
52D1 323F53     01660         LD    (OUFG),A       ;save flag
52D4 B7         01670         OR    A
52D5 2009       01680         JR    NZ,NBUFCL      ;no, no buff clr
52D7 E5         01690         PUSH  HL
52D8 0600       01700         LD    B,0
52DA            01710 CLOBFR  EQU   $
52DA 3600       01720         LD    (HL),0         ;clear buffer
52DC 23         01730         INC   HL
52DD 10FB       01740         DJNZ  CLOBFR
52DF E1         01750         POP   HL             ;buffer start
52E0            01760 NBUFCL  EQU   $
52E0 01E0FF     01770         LD    BC,-32         ;room for fcb
52E3 09         01780         ADD   HL,BC
52E4 22C953     01790         LD    (FCAD1),HL     ;fcb 1
52E7 22D853     01800         LD    (FCAD2),HL     ;fcb 2
52EA 54         01810         LD    D,H            ;save fcb addr
52EB 50         01820         LD    E,L
52EC 01D4FF     01830         LD    BC,-OTL        ;code length
52EF 09         01840         ADD   HL,BC
52F0 36C9       01850         LD    (HL),0C9H      ;set RET instr
52F2 22FE53     01860         LD    (CODA),HL      ;save code addr
52F5 010400     01870         LD    BC,CNLSB-OUTCOD ;offset for count
52F8 09         01880         ADD   HL,BC
52F9 22DE53     01890         LD    (CTLS1),HL     ;save addr
52FC 010800     01900         LD    BC,CNMSB-CNLSB ;offset to msb
```

```
52FF 09          01910          ADD     HL,BC
5300 22EB53      01920          LD      (CTMS1),HL      ;save count addr
5303 E1          01930          POP     HL              ;file name
5304             01940 XFLNMA   EQU     $
5304 CD1D54      01950          CALL    CHCHTY          ;check type
5307 3005        01960          JR      NC,SVEON        ;deliniter
5309 12          01970          LD      (DE),A
530A 23          01980          INC     HL
530B 13          01990          INC     DE
530C 18F6        02000          JR      XFLNMA          ;xfer next
530E             02010 SVEON    EQU     $
530E 3E03        02020          LD      A,03H           ;end char
5310 12          02030          LD      (DE),A
5311 E5          02040          PUSH    HL              ;save pos in cmd
5312 223454      02050          LD      (NBUFIN),HL     ;next move start
5315 2A4940      02060          LD      HL,(HIMEM1)     ;old himem
5316             02070 HMA5     EQU     $-2
5318 01A4FE      02080          LD      BC,-256-32-OTL-16 ;used bytes
531B 09          02090          ADD     HL,BC
531C 224940      02100          LD      (HIMEM1),HL     ;save new himem
531D             02110 HMA4     EQU     $-2
531F 23          02120          INC     HL              ;to entry byte
5320 EB          02130          EX      DE,HL
5321 2A3754      02140          LD      HL,(NBUFOT)     ;next in out buff
5324 7A          02150          LD      A,D             ;addr msb
5325 CD0354      02160          CALL    CNVEXH          ;conv to ext hex
5328 7B          02170          LD      A,E
5329 CD0354      02180          CALL    CNVEXH          ;convert lsb
532C 3648        02190          LD      (HL),'H'        ;set type = hex
532E 23          02200          INC     HL
532F 3620        02210          LD      (HL),' '        ;deliniter
5331 23          02220          INC     HL
5332 223754      02230          LD      (NBUFOT),HL     ;next cmd out
5335 ED5BC953    02240          LD      DE,(FCAD1)      ;fcb addr
5339 210000      02250          LD      HL,0            ;buffer addr
533A             02260 BUFRAD   EQU     $-2
533C 0600        02270          LD      B,0             ;rec len
533E 3E00        02280          LD      A,0             ;flag output
533F             02290 OUFG     EQU     $-1
5340 B7          02300          OR      A               ;check input
5341 2041        02310          JR      NZ,INPUFL       ;yes, open input
5343 CD2444      02320          CALL    OPNOPT          ;open file
5346 2817        02330          JR      Z,FLIFND        ;file found
5348 FE18        02340          CP      24              ;check not in directory
534A 2057        02350          JR      NZ,JDSERM       ;no, error
534C CD2044      02360          CALL    OPNOPT          ;try open output
534F 2052        02370          JR      NZ,JDSERM       ;not found
5351 010002      02380          LD      BC,512          ;0 records
5354 CD4244      02390          CALL    POSFCB          ;position fcb
5357 CD4B44      02400          CALL    ALLOCF          ;allocate space
535A CD3F44      02410          CALL    POSBEG          ;to file begin
535D 1808        02420          JR      TRYWRT          ;try writing file
535F             02430 FLIFND   EQU     $
535F 13          02440          INC     DE              ;to flag byte
5360 13          02450          INC     DE
5361 1A          02460          LD      A,(DE)
5362 F6C0        02470          OR      0C0H            ;set no file extend
5364 12          02480          LD      (DE),A
5365 1B          02490          DEC     DE
5366 1B          02500          DEC     DE
5367             02510 TRYWRT   EQU     $               ;write file sectors
5367 010000      02520          LD      BC,0            ;sector count 0
536A             02530 WRNWSE   EQU     $
536A CD3944      02540          CALL    WRTSEC          ;write sector
536D 2006        02550          JR      NZ,EFILIN       ;no more room
536F 03          02560          INC     BC              ;count sectors
5370 78          02570          LD      A,B             ;check max
5371 FE02        02580          CP      2               ;max 512 sectors
5373 20F5        02590          JR      NZ,WRNWSE       ;no, write next
5375             02600 EFILIN   EQU     $
5375 78          02610          LD      A,B
5376 B1          02620          OR      C
5377 3E21        02630          LD      A,33            ;no space
5379 2828        02640          JR      Z,JDSERM        ;error
537B ED43E353    02650          LD      (CNMSB),BC      ;save sector count
537F CD2844      02660          CALL    CLSFIL          ;close file
5382 0600        02670          LD      B,0             ;rec len
5384             02680 INPUFL   EQU     $
5384 CD2444      02690          CALL    OPNIPT          ;open file
5387 201A        02700          JR      NZ,JDSERM       ;dos error
5389 ED5BFE53    02710          LD      DE,(CODA)       ;addr for code
538D 210753      02720          LD      HL,OUTCOD       ;assume output
5390 3A3F53      02730          LD      A,(OUFG)        ;flag for out
5393 B7          02740          OR      A
5394 2803        02750          JR      Z,MVOTCD        ;move code
5396 21C853      02760          LD      HL,INPCOD       ;else input code
5399             02770 MVOTCD   EQU     $
5399 012C00      02780          LD      BC,OTL          ;length for move
539C EDB0        02790          LDIR
539E E1          02800          POP     HL              ;cmd line addr
539F D1          02810          POP     DE              ;type addr
53A0 C3A552      02820          JP      RPNEXT          ;find next entry
53A3             02830 JDSERM   EQU     $
53A3 E67F        02840          AND     7FH             ;set dead end
53A5 C30944      02850          JP      DSERRM          ;dos error mesg
53A8             02860 SENRBY   EQU     $
53A8 CD1D54      02870          CALL    CHCHTY          ;check next char
53AB 23          02880          INC     HL
53AC 38FA        02890          JR      C,SENRBY        ;normal char
53AE 2B          02900          DEC     HL              ;else back to char
53AF 20F7        02910          JR      NZ,SENRBY       ;not eoline
53B1             02920 SENROU   EQU     $               ;send route cmd
53B1 CD3154      02930          CALL    XFCMLN          ;xfer cmd line
53B4 2A3754      02940          LD      HL,(NBUFOT)     ;next out
53B7             02950 SETMBY   EQU     $
53B7 360D        02960          LD      (HL),0DH        ;terminate
53B9 2B          02970          DEC     HL
53BA 7E          02980          LD      A,(HL)          ;previous byte
53BB FE20        02990          CP      ' '             ;remove spaces
53BD 28F8        03000          JR      Z,SETMBY
53BF 216954      03010          LD      HL,RTECMD       ;command to send
53C2 CD6744      03020          CALL    DISPMS          ;displ route cmd
53C5 C30544      03030          JP      EXDOSC          ;execute DOS command
53C8             03040 INPCOD   EQU     $               ;code for disk input
53C8 110000      03050          LD      DE,0            ;fcb addr
53C9             03060 FCAD1    EQU     $-2
53CB CD1300      03070          CALL    13H             ;get char
53CE C8          03080          RET     Z               ;no error
53CF 21D4FF      03090          LD      HL,-OTL         ;len of entry
53D2 19          03100          ADD     HL,DE           ;to first byte
53D3 36C9        03110          LD      (HL),0C9H       ;set to RET
53D5 AF          03120          XOR     A
53D6 C9          03130          RET                     ;else no inp
53D7             03140 OUTCOD   EQU     $               ;out to file
53D7 110000      03150          LD      DE,0            ;fcb addr
53D8             03160 FCAD2    EQU     $-2
53DA 3E00        03170          LD      A,0             ;count lsb
53DB             03180 CNLSB    EQU     $-1
53DC 3D          03190          DEC     A
53DD 32D853      03200          LD      (CNLSB),A       ;save count
53DE             03210 CTLS1    EQU     $-2
53E0 200B        03220          JR      NZ,OUCBY        ;output byte
53E2 210000      03230          LD      HL,0            ;msb count
53E3             03240 CNMSB    EQU     $-2
53E5 2B          03250          DEC     HL              ;decr count
53E6 7C          03260          LD      A,H
53E7 B5          03270          OR      L
53E8 2813        03280          JR      Z,ENOVFB        ;no more room
53EA 22E353      03290          LD      (CNMSB),HL      ;save new count
53ED             03300 CTMS1    EQU     $-2
53ED             03310 OUCBY    EQU     $
53ED 210943      03320          LD      HL,PRVDRV       ;previous drive
53EE             03330 PDRA     EQU     $-2             ;prev drive store
53F0 46          03340          LD      B,(HL)          ;get drive
53F1 79          03350          LD      A,C             ;char to output
53F2 CD1B00      03360          CALL    1BH             ;output
53F5 2006        03370          JR      NZ,ENOVFB       ;error on write
53F7 78          03380          LD      A,B             ;previous drive
53F8 BE          03390          CP      (HL)            ;same as new
53F9 C45B44      03400          CALL    NZ,SELDRV       ;select drive if not
53FC C8          03410          RET     Z               ;same, exit
53FD             03420 ENOVFB   EQU     $
53FD 210000      03430          LD      HL,0            ;code begin
53FE             03440 CODA     EQU     $-2             ;code addr
5400 36C9        03450          LD      (HL),0C9H       ;set to ret
5402 C9          03460          RET                     ;exit
002C             03470 OTL      EQU     $-OUTCOD        ;code length
5403             03480 CNVEXH   EQU     $               ;convert to hex
5403 F5          03490          PUSH    AF              ;save char
5404 CB1F        03500          RR      A               ;msd to lsd
5406 CB1F        03510          RR      A
5408 CB1F        03520          RR      A
540A CB1F        03530          RR      A
540C CD1054      03540          CALL    CNVHEX          ;conv nibble
540F F1          03550          POP     AF
5410             03560 CNVHEX   EQU     $
5410 E60F        03570          AND     0FH             ;remove rest
5412 F630        03580          OR      30H
5414 FE3A        03590          CP      3AH             ;check num
```

```
5416 3802      03600          JR      C,SVHXCH
5418 C607      03610          ADD     A,7             ;else to alpha
541A           03620 SVHXCH   EQU     $
541A 77        03630          LD      (HL),A          ;to output
541B 23        03640          INC     HL
541C C9        03650          RET
               03660 ;check character type
541D           03670 CHCHTY   EQU     $
541D 7E        03680          LD      A,(HL)
541E FE0D      03690          CP      0DH
5420 C8        03700          RET     Z               ;eoline
5421 FE2C      03710          CP      ','
5423 23        03720          INC     HL
5424 2809      03730          JR      Z,EXINOC        ;comma, bypass
5426 FE20      03740          CP      ' '
5428 2B        03750          DEC     HL
5429 37        03760          SCF                     ;carry not delim
542A C0        03770          RET     NZ              ;not space
542B           03780 BPSPAC   EQU     $
542B 23        03790          INC     HL
542C BE        03800          CP      (HL)            ;still space
542D 28FC      03810          JR      Z,BPSPAC        ;yes
542F           03820 EXINOC   EQU     $
542F B7        03830          OR      A
5430 C9        03840          RET
5431           03850 XFCMLN   EQU     $               ;xfer to cmd line buffer
5431 D5        03860          PUSH    DE
5432 E5        03870          PUSH    HL
5433 010000    03880          LD      BC,0            ;prev moved
5434           03890 NBUFIN   EQU     $-2
5436 110000    03900          LD      DE,0            ;prev dest
5437           03910 NBUFOT   EQU     $-2
5439 B7        03920          OR      A
543A ED42      03930          SBC     HL,BC           ;# bytes to move
543C 280E      03940          JR      Z,XFCMEX        ;zero len move
543E 44        03950          LD      B,H             ;len to bc
543F 4D        03960          LD      C,L
5440 2A3454    03970          LD      HL,(NBUFIN)     ;prev move end
5443 EDB0      03980          LDIR                    ;else xfer
5445 223454    03990          LD      (NBUFIN),HL     ;next to move
5448 ED533754  04000          LD      (NBUFOT),DE     ;next dest
544C           04010 XFCMEX   EQU     $
544C E1        04020          POP     HL
544D D1        04030          POP     DE
544E C9        04040          RET
               04050 ;
               04060 ;type table
544F           04070 TTBL3    EQU     $               ;mod 3 table
544F 52        04080          DEFM    'RI'            ;rs232 input
     49
5451 01        04090          DEFB    1
5452 E541      04100          DEFW    41E5H
5454 52        04110          DEFM    'RO'            ;rs232 output
     4F
5456 00        04120          DEFB    0
5457 ED41      04130          DEFW    41EDH
5459           04140 TTBL1    EQU     $               ;mod 1 table
5459 4B        04150          DEFM    'KB'            ;keybd input
     42
545B 01        04160          DEFB    1
545C 1540      04170          DEFW    4015H           ;addr of fcb
545E 44        04180          DEFM    'DO'            ;display output
     4F
5460 00        04190          DEFB    0
5461 1D40      04200          DEFW    401DH
5463 50        04210          DEFM    'PR'            ;print output
     52
5465 00        04220          DEFB    0
5466 2540      04230          DEFW    4025H           ;fcb addr
5468 00        04240          DEFB    0
5469 52        04250 RTECMD   DEFM    'ROUTE '
     4F 55 54 45 20
0048           04260 RTEBUF   DEFS    75
0100           04270          DEFS    256             ;room for stack
55BA           04280 STAK     EQU     $
5200           04290          END     BEGIN
00000 TOTAL ERRORS
```

```
ALLOCF 444B   BEGIN  5200   BPSPAC 542B   BUFRAD 533A   CHCHTY 541D
CHKRT  525B   CLOBFR 52DA   CLROUX 52A3   CLSFIL 4428   CMLSB  5308
CMNSB  53E3   CMVEXH 5403   CMVHEX 5418   CODA   53FE   CTLS1  530E
CTHS1  53EB   DISPHS 4467   DOSCMD 4419   DSERRH 4409   EFILIN 5375
ENDVFB 53FD   EXDOSC 4405   EXINOC 542F   FCAD1  53C9   FCAD2  5308
FLIFND 535F   FNEOSN 52A8   FNTYEN 5236   HIMEM1 4049   HIMEM3 4411
HMA1   528E   HMA2   529E   HMA3   52C9   HMA4   531D   HMA5   5316
HMAP   5294   INMOD1 5232   INNEXH 52A7   INPCOD 53CB   INPUFL 5384
JDSERH 53A3   LVHOME 52A0   MVOTCD 5399   NBUFCL 52E0   NBUFIN 5434
NBUFOT 5437   OPNIPT 4424   OPNOPT 4420   OTL    002C   OUCBY  53ED
OUFG   533F   OUTCOD 5307   OVLFG1 4369   OVLFG3 4289   PORA   53EE
POSBEG 443F   POSFCB 4442   PRVDR3 427E   PRVDRV 4388   ROUTAS 4833
RPNEXT 52A5   RTEBUF 546F   RTECMD 5469   SELDRV 445B   SENRBY 53A8
SENROU 53B1   SETMBY 53B7   STAK   55BA   SVEDN  530E   SVHXCH 541A
TRYMRT 5367   TTBL1  5459   TTBL3  544F   WRNWSE 536A   WRTSEC 4439
XFCMEX 544C   XFCMLN 5431   XFLNMA 5304
```

SWAPPING AND SHARING

by Jim Butterfield, TORPET September '83

[Although the name Jim Butterfield may not be familiar to TRS-80 users, Jim is considered something of a software "guru" among Commodore users. He is one of the founding fathers of the Toronto PET Users Group (TPUG). Those of our readers that belong to user groups will find this article thought-provoking. It was reprinted from the September, 1983 issue of the TORPET, which was at that time (but is no longer) the newsletter of TPUG.]

I must confess that I can't understand the logic of swapping programs.

Sure! you have a spare cat you don't need, and your friend has a shoe polishing kit... go ahead and swap, you'll both benefit. But programs are different.

I can see the situation where each of the two parties have written a program. You've written a telephone list, and I've written a simple game... why not swap?

But even then, it flies in the face of good sense.

You can give away a program -- and still have it. If it's yours - or if it's public domain - you incur no loss. Maybe, as the saying goes, he who steals my purse steals trash... but I'm out one purse. On the other hand, he (or she) who gets a copy of my program may also get trash... but I have lost nothing.

Occasionally, I run across someone who has an attractive program. And when I ask, "Is that public domain? May I have a copy?", I get the reply, "What can you swap me for it?" My answer: "Nothing. All my programs are in the TPUG library". So I don't get a copy of the program.

This amazes me. The other person may have dozens - or hundreds - of my programs. But I'm not going to get the new program, because I have nothing to swap.

A few years ago, I received a letter from Oregon, asking if I had any music programs. The writer had bought a commercial package and interface, but didn't have much music. I put together a cassette of all the music I had... a dozen programs or so.

About a month later, a letter came from northern California. It said, "I got a copy of your music programs from XYZ in Oregon. I have some music programs of my own. What programs do you have to swap me for them?" Again, I had to reply, "None - I sent them all to Oregon, you have them all now."

The whole swapping thing makes no sense to me. The name of the game is sharing, not swapping.

Let's look back at the origins of the club. Suppose I - and several other programmers - had said to TPUG, "You don't get programs from us unless you can swap us something equally good". Suppose that TPUG said to its members, "You don't get a program until you submit a program of equal quality". We'd have a pretty weak operation. User groups don't work that way. Thank heavens!

I fear that the swap syndrome encourages program theft.

Some poor beginner who isn't skilled in program writing is coerced by swappers into giving a program as a swap. What is he or she going to give? The pressure is to buy a program and give away a copy. And that's wrong, wrong, wrong.

Sometimes I send people programs. I usually refer them to the club, but occasionally I need to send a program or two directly. I don't expect any in return; in fact, sometimes my return address is not on the package. Some people reply and say "Thank you!", which is OK. On a couple of occasions people have replied by sending me bootleg copies of commercial programs. They shouldn't do that. I have a feeling that these people have been brainwashed into the "swapping" thing. They think that they must give something in return - even if it's illegal. They shouldn't.

Let's get off this swapping bandwagon.

Any programs I have, provided they are not copyrighted or commercial, are freely available to anyone who wants them. They are in the club library, for that matter.

How about your programs? Surely you don't think that they are too good for the club? Throw them into the pot - make them available.

The whole business of having a club is to share ideas, experiences - and programs. Let's share -- not swap.

MAGIC MATH PLUS
A new series feature in Northern Bytes
by Dr. Michael W. Ecker

[Dr. Michael Ecker, an Associate Professor of Mathematics and Computer Science at the University of Scranton, is a contributing editor of Popular Computing, Byte, and Soft Sector, for which computer magazines he writes columns on mathematical and computer recreations. Most notable are his monthly column "Recreational Computing" in Popular Computing and a bimonthly column "Mathematical Recreations" in Byte. A book based on material similar to that in his columns is scheduled for release by Arco Publishing in the Fall of 1985. He also does occasional freelance pieces on financial mathematics, and software reviews. As the alter ego of Recreational Mathemagical Software, Dr. Ecker offers this serialization of some of his Magic Math Plus software, a collection of programs of "mathemagic", games, educational programs, financial utilities and numerous number tricks. At least one program from Magic Math will be offered in each installment. Readers who would prefer not to type in programs can purchase a disk directly from him. Northern Bytes is pleased to extend its coverage of useful information for TRS-80s to basic Basic applications and recreations. Information regarding questions for Dr. Ecker and/or ordering will be provided at the end of each article.]

Fastloan! A Quick Loan Amortization Program (with loan amortization table)

Psst! Want to save yourself $20,000 in interest on a home mortgage loan? I did. Read on to see how this loan amortization program gives a quick picture of what you'll pay under various time periods and even with slightly different interest rates which may be available.

If you're anything like me, you probably have accommodated yourself to borrowing as a part of modern life. Besides having a loan in the form of a mortgage, I had so little saved that I even had to use my credit cards to get a cash advance to pay the down payment and closing costs! Needless to say, I wished to have a utility on hand at any time to analyze a loan. I wished it to be quick and easy to use - even without instructions. Just pop in the amount of the loan (without dollar signs or commas), the annual interest rate (omitting the "%" sign, using 14 for 14%, for instance), and the term of the loan in years.

Most people do not grasp the numerical realities of loans. Since it is not that difficult for a mathematician to computerize the process, why not make it available not just for my own use, but for others as well? Thus, I created Fastloan. I designed it without a lot of flourishes, but I wanted a "what if" kind of feature found on spreadsheets such as VisiCalc and its many mimics. I settled on the ability to hit the ENTER key to avoid retyping the same information. Of course, you can change any combination of the values that you try, keeping some the same or changing them all. Fact is, all the instructions are already included within the program anyway. You can print out an amortization table; i.e. a table showing your monthly payments, how much goes to interest, to principal, and the remaining balance. It should be very eye-opening for many. But the main thing is you can play around, experimenting with different combinations of interest rates and repayment periods. You will be surprised, for instance, to see that repaying a home mortgage on a $50,000 home five years earlier can save you tens of thousands of dollars in interest!

I leave you to type in and play (or is it work?) with the program.

```
1 CLS:PRINT:PRINT:PRINT CHR$(23)"
FASTLOAN!":PRINT:PRINT:FOR DELAY=1 TO 1500:NEXT:CLS
2 PRINT:PRINT "                    FASTLOAN!
                                   Dr. Michael W. Ecker
                                   129 Carol Drive
                                   Clarks Summit, PA 18411"
3 PRINT:PRINT:PRINT:INPUT "<ENTER> TO
CONTINUE";Q$:CLS:PRINT "     THIS PROGRAM GIVES MONTHLY
PAYMENT AMOUNTS ON LOANS."
4 PRINT "
==================================================="
5 FOR DELAY=1 TO 500:NEXT
8 PRINT:PRINT
9 PRINT " -->  FOR DOLLAR ENTRIES, DO NOT USE DOLLAR
SIGN OR COMMA.":FOR DELAY=1 TO 500:NEXT
10 PRINT:PRINT " -->  FOR AN INTEREST RATE OF, SAY, 12.5%,
USE 12.5"
11 PRINT "      (Don't use .125 or 12.5% or other such variations.)"
15 PRINT
17 FOR DELAY=1 TO 500:NEXT
18 PRINT
20 INPUT "PRINCIPAL (AMOUNT BORROWED)";P
30 INPUT "ANNUAL INTEREST RATE (percent)";I
35 R=I/100
40 INPUT "HOW MANY YEARS";Y
50 A=(P*R)/(12*(1-(12/(12+R))[(12*Y)))
85 PRINT:PRINT "THE MONTHLY PAYMENT IS"
90 PRINT USING "$###.##";A
95 PRINT "TOTAL PAID = ";:PRINT USING "$###,###";12*Y*A;
97 PRINT "    INTEREST PAID = ";:PRINT USING
"$###,###";12*Y*A-P
100
PRINT:PRINT"**********************************************
***************"
105 PRINT " FOR A LOAN AMORTIZATION TABLE, TYPE 'A' AND
<ENTER>."
110 PRINT " FOR DIFFERENT VALUES, JUST HIT <ENTER>
without the 'A'."
115 PRINT " NOTE: WHEN YOU ARE ASKED QUESTIONS AGAIN, IF
YOU"
116 PRINT " HIT <ENTER>, THEN COMPUTER WILL USE SAME
VALUE AS"
117 PRINT " PREVIOUSLY USED FOR THAT PARTICULAR
QUESTION."
120 INPUT X$:CLS
127 IF X$="A" THEN X$="":GOTO 200
130 PRINT:PRINT:GOTO 20
200 PRINT "          ..... AMORTIZATION TABLE ....."
202 PRINT "           based on a monthly payment of ";
203 PRINT USING "$###.##";A
204 FOR DELAY=1 TO 1000:NEXT:PRINT
206 PR=P
208 FOR YY=1 TO Y
210 PRINT "PAYMENT #", "INTEREST","PRINCIPAL","BALANCE
OWED"
215 PRINT
230 FOR J=1 TO 12
235 I1=PR*R/12
238 PRINT 12*(YY-1)+J,
240 PRINT USING "###.##";I1,
241 PRINT,:PRINT USING "###.##";A-I1,
242 PRINT,:PRINT USING "#####.##";PR-(A-I1)
245 PR=PR-(A-I1)
248 NEXT J
250 PRINT:INPUT "    <ENTER> to continue...";AA$
260 NEXT YY
265 IF Y>INT(Y) THEN GOSUB 400
275 PRINT:PRINT "    THAT COMPLETES THE AMORTIZATION
SCHEDULE."
280 PRINT:PRINT:INPUT "<ENTER> TO CONTINUE";AD$
300 CLS
310 PRINT "TO RESTART, HIT <ENTER>."
320 INPUT AA$:CLS
330 GOTO 20
399 END
400 ' *** SUBROUTINE PATCH FOR NONINTEGRAL NUMBER OF
YEARS ***
410 PRINT:PRINT
420 L=Y-INT(Y)
430 FOR MO=1 TO 12*L
440 I1=PR*R/12
450 PRINT 12*INT(Y)+MO,
460 PRINT USING "#,###.##"; I1,
470 PRINT ,: PRINT USING "#,###.##"; A-I1,
480 PRINT ,: PRINT USING "#,###,###.##"; PR-(A-I1)
490 PR=PR-(A-I1)
500 NEXT MO
510 RETURN
```

I actively solicit your comments, pertinent questions, suggestions, improvements and so on. Write to me directly at the address above. You may also obtain a copy of Magic Math Plus (copyright 1984, Michael W. Ecker, PhD) for the TRS-80 Models 3, 4, 4P with at least one disk drive. Volume 1 contains approximately 20 programs, including Fastloan, Compound Interest, Super-blackjack, Super-Trick, Super-Fast Prime Number Cruncher, The Game of N, Fibonacci Numbers, Additive Sequences, Base Two (a trick with explanation) and loads of other goodies!

Magic Math comes on a self-booting disk with XDOS, a licensed disk operating system compatible with TRSDOS 1.3, from Mr. Jim Kyle of the Software Factory. The collection is totally menu generated (again, courtesy of software from the Software Factory and available from Recreational Mathemagical Software). This combination makes use of Magic Math almost automatic.

Volume 1 sells for $24.95 plus $1 for shipping and handling. A volume 2, written by programmer/writer Mr. David B. Lewis (who

writes in the likes of 80 Micro and others), is available with lots of other numerical curiosities involving convergence, graphic and other games (as in Hexapawn, Repeater, etc.), with at least 15 programs available for the same price. Same menu generated format available for Volume 2.

Special offer: Volumes 1 and 2 combined, with the self-booting disk, XDOS, a minimum of 36 programs (including some bonuses and tutorials) in a menu generated format, all for $36 ($1 per program) plus $1 shipping and handling.

Lastly: I will be pleased to send anybody who wishes further information a catalog I am preparing if you send a self-addressed stamped envelope. I remind you again that your comments and questions are welcome and appreciated. Until next issue! Happy Computing!

Dr. Michael W. Ecker
Recreational Mathemagical Software
129 Carol Drive
Clarks Summit, PA 18411
(717) 586-2784

DOTWRITER/ALLWRITE!
by Dave Bower

I'm a dedicated Allwrite!/Dotwriter user. (Allwrite! is by Chuck & Glenn Tesler, Dotwriter by W. K. Mason and both are distributed by PROSOFT.) I firmly believe they can do just about anything, it's just a matter of figuring out how to do it.

Some of the areas that gave me problems with Allwrite! were super/subscripting in the proportional mode with an Epson FX-80 (PROSOFT & Epson said it can't be done), formatting my text for transmitting over the phone lines and key clicks (no problem with the clicks, just a stupid mistake on my part).

In Dotwriter I'm going to talk about putting more than two fonts on the same line and the .PA n command.

You can't sub/superscript in the proportional mode on the Epson FX-80. But you can do it in any other mode. So print your text in proportional and your scripting in another pitch.

You can do this because changing pitches in Allwrite! does not cause a control break. You can print each letter in a word in a different pitch, if that's what rings your bell. For example to print the word, H2O with the 2 subscripted you would have the following lines in All Write!

```
. . . the formula for water is H
¡pi 16
@-2@=
¡pi 0
O and the formula of . . .
```

Condensed is the best choice for scripting with the proportional mode. Condensed is not proportional, but it is small enough not to be noticeable.

Another problem is setting up a file to transmit over the phone lines. Allwrite! does not add carriage returns at the end of each line. You could hit an <ENTER> at the end of each line, but then you lose the benefit of Allwrite's formatting capabilities. The following steps will add a carriage return to the end of each line. Do it after you've finished composing your text and are ready to send it.

1) Change every double carriage return to a <CR><####><CR>

2) Create a soft key (use X, Y or Z because they can repeat) to go to the end of the line (shift/right arrow) & add a carriage return. <SHIFT/RIGHT ARROW><CR>

3) Use your newly created soft key (X, Y or Z can be used up to 32 times at one shot) to change your text.

4) Go back and delete the ####'s in your text.

The reason for the #### is to prevent any line from having only a carriage return. Your soft key will skip the line following a line with only a carriage return.

As a newsletter editor I constantly begged people to send in the solutions to the "little" problems they've overcome. Because A) they're not little at the time they are hanging you up and B) no man is an island; if something caused confusion for you, the same something caused confusion for, at the very least, many others!

So here's my stupid mistake of the week. I could not get the Model 4 Allwrite! to give me key clicks. I don't know how many times I ran the installation program, before I finally read the question. The Model 4 Allwrite! doesn't supply key clicks, you must use the DOS to configure the system for clicks. In TRSDOS 6 you do this with the following commands.

```
SET *SP CLICK/FLT
FILTER *KI *SP
SYSGEN (YES)
```

Don't waste to much time on this though, the clicks on the 4P are slightly lower than the threshold of human perception.

Let's talk Dotwriter.

Do you know that you can specify the number of the next page when you use the page eject (.PA) command? I got a call from a friend who was printing a book and wanted to leave large groups of blank pages (10 to 20) throughout but didn't want to print a blank page for each one. Seems like a good idea to me, but we couldn't find any instructions in the book. That's only because the book left out that part of the PAGE EJECT command. The reference card shows the command with the operand.

How about four fonts on one line? It makes a pretty spiffy title line. Use columns. Columns don't have to be in effect the whole page. In fact you can print EACH line in a different number of columns.

I needed four fonts across the top of the page. I used four columns, forcing a column and using a new font each time. I had four columns of one line each and then went to one column for the rest of the page. With some judicious spacing it looked like one line printed with four fonts. (I could have easily used my alternate font and put eight fonts on the page.)

I think Dotwriter (I am running Version 4.0) and Allwrite! are the best thing to happen to words since the introduction of paper with lines on it. Should you have any questions, comments or suggestions you'd like to share, my MCI Mail ID is 224-9092 (my user name is DBOWER).

[Editor's note - When Dave originally sent me the above article, he used the name "Dot Print" instead of "Dotwriter". I sent him a message asking about this discrepancy, and received the following reply:

"..... USUALLY it is called Dot Writer in the documentation. The title of the 3.0 manual is Dot Writer 3.0, the blurb on the first page uses Dotwriter twice and the page titles are Dotwriter while within in the text of the manual they refer to Dotprint.

"The title of the 4.0 manual is Dotwriter 4.0, text in the 4.0 manual refers to Dotwriter, but page headings are Dot Writer. Within the manual they refer to it as Dotprint. The word Dotwriter (without the space) is a trademark of W. K. Mason, the author of Dot Writer.

"Dotwriter is the name of the package, while Dotprint is the name of the program that prints it. They are one and the same. The package is the program. You must use any other word processor to write your text and Dotprint will print it....."

Well, PROSOFT's advertising refers to the package as "Dotwriter", so I changed the name within Dave's article to conform to that usage. It would seem this is one product with a bit of an identity crisis!]

THE PRESIDENT'S MESSAGE
by Gary DiIllio

[Reprinted from HARDCOPY, newsletter of the TRS-80 User's Group of Delaware County]

This month I would like to present Gary's Rules of Computer Advice. We microcomputer owners are often asked to advise someone on the purchase of a micro. If, after reading these rules, you come to the conclusion that advising consumers on microcomputers is a "no-win" situation, do not blame me!

GARY'S RULES OF COMPUTER ADVICE:

1 - At least half of those seeking advice on selecting a microcomputer have already decided on a particular brand of micro BEFORE they ask you for advice.

2 - About 10% of those seeking advice on selecting a micro BOUGHT one last week. They are not looking for advice; only encouragement that they made the right decision.

3 - If someone heeds your advice, you will be cursed and blamed every time "SYNTAX ERROR" appears on the screen.

4 - If someone ignores your advice you are justified in getting angry.

5 - Anyone who works with or owns a micro is an "expert". After having worked, for 8 years, with over 20 computer systems from the largest to the smallest, I enjoy being told that "Harry the barber just got an APPLE last week and he says...".

6 - People who wouldn't think of comparing a $10.00 transistor radio to a $3000.00 stereo multiplex system will want you to compare and contrast a $50.00 TIMEX computer with an $11,000.00 APPLE LISA; don't do it!

7 - Often you are asked to advise on selection of a micro after the following exchange:
How much do you want to spend? "Not much!"
What do you want to do with the micro? "This and that!"
What particular kind of software do you want to run? "All kinds."

8 - After spending an hour trying to fit a system to the purpose, then announcing the decision don't be surprised to hear "But the color clashes with my drapes!"

ABOUT VIVACE....
by Lou Witt

[We don't normally use this type of article in Northern Bytes, since it could be considered advertising for a particular software product (not to mention that it's a product sold by The Alternate Source!). After all, folks might consider it a sneaky form of advertising. In this case, though, I actually requested permission to print this in Northern Bytes, because I think it provides some good insights into how BASIC compilers work, and documents the limitations of many compilers. After reading this, you'll have a better idea of the tradeoffs involved in the design of a BASIC compiler (compatibility with TRS-80 BASIC vs. execution speed of the compiled program). So, although this article may have been written to help sell Vivace!, I believe it has sufficient educational value to warrant reprinting here. I hope you agree!]

VIVACE (vee-VAH-chaa): Lively, vivaciously, with vigor. [from the Latin 'Vivus'- living, alive.]

WittSoft's Vivace! is a unique product. With it you may convert practically any Basic program to a /CMD program. This allows faster execution, code security, and ease of use. It is the first (and only!) system that does not require editing of source programs prior to compiling.

We've had several requests for a comparison of Vivace!'s features as opposed to similar systems. The following was taken from a letter written by Vivace!'s author. As with any program the end user must determine which particular title will fill his needs best. We hope this aids in that decision making process.

In November of '83, WittSoft sent out a mailing in which we requested input on what kind of programs our customers wanted. The one thing that come up several times was a 100% compatible Basic compiler that was easy to use and didn't require vast amounts of memory or disk space. It seems there are many users who don't want to re-write their existing programs to fit the many limitations of existing compilers.

Compatibility (including memory and disk size) were therefore the primary goals in the creation of Vivace!. Ease of use was the secondary goal, and finally enhanced execution speed. At this point you may say, "But Speed is the most important goal in a compiler!". In some cases that is true, and the compilers that are presently sold seem to have taken that as their primary concern. There are users of Basic who don't consider that as the sole consideration in their choice of a compiler, and it is for those users that Vivace! was written.

In this world there is no such thing as a free lunch. There are only two ways to make a program run faster. One is to eliminate features (i.e. it executes differently after compiling). The second is to implement a more efficient way of doing EXACTLY the same thing (it executes more efficiently after compiling). BASCOM, Accel, Zbasic, and WIBasic use the former approach. Vivace! the latter.

The obvious effect of eliminating features is the deletion of keywords from the Basic instruction set. This is the obvious effect, there is a second effect which is hidden from the user. With the exception of Vivace! all compilers alter the way programs execute. An example of this is the use of a pause (shift key during execution. Vivace! compiled programs will pause if you press that key. The others won't. The same is true of the Break key. You can terminate a Vivace! compiled program by pressing Break.

You can learn to live without pause and break in a program, but other changes are far more significant, particularly the structure of the source program.

Other systems require that "definition elements" (such as DEFtype, DIM, and FOR/NEXT) be statically structured and defined (the exact limitations will be defined a little later). If you're writing a brand new program this (like the special key problem just mentioned) is no great hardship. But what about the programs you already have? What about the 28K accounting package that doesn't have any remark statements? Re-structuring these programs to fit the compiler's limitations is time consuming at best and in some cases impossible. Consider the following:

```
10 CLEAR 0: X=(FRE(0)-1000)/64: CLEAR X
20 X=FRE("")/64: DIM TX$(X)
```

This block of code is taken from a Basic text editor. Line 10 determines how much memory is available in the system, then sets aside the maximum amount of string space. Line 20 dimensions an array to use the space. This code is acceptable to Vivace! and will perform as intended. No other system will compile this code.

Let's consider the case of a program which dynamically sets the default types of variables.

```
10 INPUT "WILL YOU BE USING NUMBERS > 999,999 (Y/N)";A$
20 IF A$="Y" THEN DEFDBL R ELSE IF A$="N" THEN DEFSNG R
30 IF A$<>"Y" AND A$<>"N" THEN 10
```

Vivace! will allow this kind of dynamic type definition. The others may not signal an error, but will define the type as SNG regardless of the user's input.

That's a rather obvious example, but a hidden effect will often introduce errors into a program that runs fine under the interpreter. Standard Basic interprets the result type of an expression as each element is evaluated. This allows for maximum speed by doing operations on integers in registers, yet allows for the automatic conversion of an intermediate result to a higher precision if required by the values involved.

```
10 A%=18157: B%=27398: C%=22777
20 D%=(A%+B%)/2
30 PRINT C% " = " D%
```

This will work under Vivace! and conventional Basic, but will generate an erroneous result under most compilers. Most compilers force the result of "(A%+B%)" to integer type. This overflows the integer limit, and causes an (erroneous) negative result. Standard Basic and Vivace! detect the overflow and (correctly) treat the intermediate value as a single precision quantity.

There are quite a few other limitations that aren't readily apparent. The only compiler that comes close to Vivace!'s compatibility level is BASCOM, so this comparison will concentrate on that product. A brief comparison, indexed by keywords follows:

CALL: BASCOM treats CALL as a reference to an external program segment developed with Microsoft Assembler. Vivace! handles it exactly the same as Basic.

CHAIN: BASCOM does not permit the ALL, MERGE, or DELETE options. It will allow COMMON, but the COMMON statements in the chained program must be identical. BASCOM will only chain to another compiled program. Vivace! allows all forms of CHAIN, and can chain to a compiled, uncompiled, or ASCII program.

CLEAR: Depending on the version, BASCOM either doesn't allow CLEAR at all, or restricts its use to nulling variables. Vivace! supports CLEAR including all options.

COMMON: BASCOM requires that the COMMON statement be the first in any program, and that all programs have identical lists of variables. Vivace! allows COMMON anyplace within the source program, and the lists may differ.

DEF FN: BASCOM requires the definition physically appear before any reference, and does not allow redefinition. In some BASCOM versions the function arguments are not local. Vivace! accepts any function definition that will run under Basic, including local argument definition and dynamic redefinition.

DEFINT,DEFSNG,DEFDBL,DEFSTR: BASCOM requires the statement physically precede any affected variable, and will not permit dynamic retyping. Vivace! allows dynamic variable typing (including placement of the DEFtype statements in a subroutine).

DIM: BASCOM will only accept constants as DIM arguments, and the DIM statement must physically precede any reference to the array. Vivace! allows constants, variables, or expressions as arguments and will accept any structure which runs under Basic.

ERASE: BASCOM will not support this command. Vivace! will.

ERROR TRAPPING: BASCOM requires special flags to indicate error trapping will be used. Vivace! supports error trapping directly.

SYSTEM: Vivace! is directly compatible with the SYSTEM command. BASCOM only allows a limited number of SYSTEM options. [SYSTEM is called CMD on the Model III version].

FOR/NEXT: BASCOM requires absolute adherence to "proper" structure in the nesting of FOR/NEXT loops. Vivace! will allow any structure which runs under the interpreter.

FRE: BASCOM allocates the entire variable table at compile time, thus FRE and MEM will not reflect the same values as would be the case under Basic. Vivace! dynamically assigns variable space, thus the FRE and MEM functions continue to work.

READ: BASCOM requires the DATA match the type of the READ variables (thus you cannot read a numeric variable into a

string). Vivace! allows any read logic that would run under Basic.

RETURN: BASCOM will abort on a return without gosub. Vivace! will generate an appropriate error (which may be trapped).

INPUT: BASCOM does not handle INPUT the same as Basic. Vivace! does.

LOAD: BASCOM does not allow any form of LOAD. Vivace! will allow LOAD with the ",R" option of compiled, uncompiled, or ASCII programs.

NEW: BASCOM chokes on NEW. Vivace! treats it as an END.

ON ERROR GOTO: BASCOM requires flags be set at compile time to specify that this statement is used. Operation is not the same as Basic. Vivace! handles this statement exactly the same as Basic.

RUN: BASCOM will only allow RUN to be to another compiled program. Vivace! allows all forms of RUN.

STOP: BASCOM terminates on the STOP command. Vivace! also terminates, but displays a "Break in ###" message first. Vivace! will also allow you to terminate a running program by pressing the Break key.

TRON/TROFF: BASCOM requires special flags at compile time to use the command. Vivace! doesn't.

USR: BASCOM allows user calls, but does not use the same parameter passing conventions as Basic. Most USR routines will have to be rewritten to function with BASCOM programs. Vivace! supports exactly the same USR conventions as Basic. In addition, Vivace! maintains a compatible variable table and pointer block, so USR routines don't have to be rewritten to work under compiled code.

VARPTR: BASCOM will return the address of variable storage, but the storage is formatted differently than Basic. i.e. A USR routine that expects to find array dimensions at VARPTR(array)-1 will crash. Vivace! uses the same table structure as Basic, thus these routines work as-is.

WHILE/WEND: BASCOM requires absolute structure on WHILE/WEND blocks. Vivace! will accept anything that runs under Basic.

AUTO, CONT, EDIT, LIST, LLIST, MERGE, RENUM, SAVE: BASCOM will generate an error if any of these command words is in a program. Vivace! simply ignores them, since they are almost always in a block that will not execute during normal runs, as in:

```
1 GOTO 3
2 SAVE "PROGRAM/BAS": STOP
3 REM
```

I personally put this in every program I write. During development I just "RUN 2" to save a current copy to disk. Vivace! assumes that you are doing something similar if it runs into a command word, and ignores it.

As you can see, most programs will require some editing to be compiled by BASCOM. Vivace! accepts practically any program.

BASCOM can expand programs up to 3 times (a 12K /BAS file becomes a 36K /CMD file). Vivace! generally expands only 10 to 20 percent (a 12K /BAS file becomes a 14K /CMD file). In terms of disk space, Vivace! generates the /CMD file only. BASCOM (depending on options) generates one or more intermediate and listing files, plus requires the source code be in ASCII.

That covers the first goal (compatibility), now for the second (ease of use).

Vivace! accepts a plain /BAS file for input and generates a /CMD file as output. You don't need to save the file in ASCII, run compiler output through a linking-loader, or interface to any library files. This is all done automatically. One of my BETA testers successfully compiled a huge inventory management program after spending about 10 minutes with the Vivace! documentation. He didn't have to change a single line of code, even though the program uses unstructured FOR/NEXT loops, CLEAR statements, and embedded command words. So far the only tested program that failed had machine code packed into remark lines.

The final goal was speed. That is an area where the type of program being compiled has a great effect. Generally, Vivace! eliminates delays due to: string garbage collection (silent death), variable searches, array searches, text searches. That means a 3 line program, which isn't being slowed down by any of these factors won't show much gain after compiling. On the other hand, short programs with few variables are NOT the ones that are slow. Large programs with substantial variable sets (the kind people really use) can show significant improvement. Users report many programs run twice as fast. I've generated programs that ran up to 10 times faster, and the string reorganization can run up to 200 times faster than Basic.

The bottom line comparison of the two is this:

BASCOM is a programmer's tool. It isn't directly compatible with Basic, requires an investment in time and energy to learn its limitations and features, and requires significant system resources (memory and disk space) to be used effectively. The user must possess Basic language programming skill in order to convert existing programs prior to compilation. Where necessary compatibility and memory efficiency are sacrificed for speed.

Vivace! is a user's tool. It is directly compatible with Basic, requires practically no investment in time or energy to use, and will operate in a typical user environment. The user need have no knowledge of the Basic programming language whatsoever. Where necessary speed has been sacrificed in favor of compatibility.

I really feel Vivace! should be in every user's library. As an experienced programmer, I can use BASCOM, WIBasic, Accel, or Zbasic to write very fast programs. With a few hours of editing I can convert some existing programs to a syntax that would allow compiling by one of these fine products.

With Vivace! practically anyone can compile practically any Basic program, and achieve a good speed increase. I have the necessary skill to use the other systems, but until Vivace! I didn't compile most of my Basic library because it would have been too much trouble. With Vivace!, I've compiled them all. No blood sweat or tears either. Just put a /BAS file in one end and get a /CMD file out the other.

If we define "Basic" as the language that runs on the TRS-80, there's only one compiler: Vivace!

Yours,
Lou Witt
(author, Vivace!)

HARDWARE PRODUCT WANTED

What the world (or at least the microcomputing world) really needs now is a good, inexpensive 1200 baud Modem. I cannot understand why this product has not already appeared on the market. You can get a good, inexpensive 300 baud Modem for around $50. In that price range they have no fancy features (they are a plain old "dumb" Modem that has maybe an on/off switch and an answer/originate switch), but they work just fine. I'm not too lazy to dial the phone myself, but I would like to be able to communicate at 1200 baud without spending nearly $300!!! In other words, I want a plain, cheap, "no-frills" 1200 baud Modem.

I hope a few of you hardware types will take up the challenge and attempt to build such a device. Here's some help: Advanced Micro Devices, 901 Thompson Place, P.O. Box 3453, Sunnyvale, California 94088 manufactures the Am7910 WORLD-CHIP™ FSK Modem IC. This "Modem on a chip" is capable of operating under Bell 103 (300 baud), Bell 202 (1200 baud), CCITT V.21 (300 baud), and CCITT V.23 (600/1200 baud) standards in any of nineteen separate modes. In other words, if you wanted to dial up an overseas BBS (which uses the CCITT standards) you could do it if you were using a Modem built around this IC. For more information on this chip, you can get the Am7910 technical manual, and an applications manual entitled "The Chip Heard Around the World", free of charge by contacting Advanced Micro Devices. You may write to them at the above address or phone (800) 538-8450 (toll-free in U.S.) or their regular number (408) 732-2400. They also have sales offices in some major cities of the U.S. and around the world, check your phone book. Also, they have a TWX number, (910) 339-9280, and a TELEX number, 34-6306. The manuals are not intended for the beginner in electronics, but if you have any electronics background you should find everything you need to know (and more!) in these manuals!

One suggestion if you decide to try and develop a commercial product around this chip - be sure to put in a switch that permits the user to select the Bell or CCITT standard. It might be tempting to leave it out and save fifty cents, but if you do you will cut off your overseas market. I envision a unit that would have four switches on the front panel: 1) On/Off, 2) Originate/Answer, 3) 300/1200 baud, 4)Bell/CCITT standard. And that's ALL! If you insist on doing something fancy, put in a circuit to monitor the phone line so that if the voltage on the line goes to zero, the Modem will shut itself off (something like an automatic disconnect on a telephone HOLD circuit, that releases the line if the party on hold hangs up). If you don't want to make a commercial product but would like to design a circuit for your own use (or for your computer club to build), why not keep notes and write it up as an article for Northern Bytes?

I really do hope someone will do something with this. Down with expensive Modems!

NOTES ON THE NEW VERSION OF TASMON

The current version numbers for TASMON as of November 1, 1984 are as follows: Model I, version 2.22. Model 3, version D7. Model 4, version 1.11. Model 4 INSTALL/CMD program, version 2.0.

Early purchasers of the TASMON upgrade (one version number below the ones shown above, i.e., 2.21, D6, and 1.10 for the Models I, III, 4 respectively) may notice an error during program disassemblies, wherein instructions such as LD L,(IX+n) are disassembled as LD LX,(IX+n). If you got one of these, you may get a free correction by sending your master TASMON disk and a stamped, self-addressed disk mailer to either TAS or NORTHERN BYTES. This ONLY applies to folks that find this particular error (relatively few were sold with this error). Second, it should be noted that the upgraded version of TASMON is longer than the original version. This means that it will NOT load into 16K machines (are there any of those still around?), and that the Model I and III cassette versions of TASMON are no longer being sold. However, if you have 16K and/or no disks, but have a friend with a disk system of the same model (I or III/4), you can buy TASMON on disk and have your friend transfer it to tape for you (using TASMON itself to make the transfer). Here's how to do it:

Execute TASMON from DOS READY.
If you have more than 16K, type:

```
W T 6000 80FF 6000 TASMON
```

If you have only 16K, type:

```
X 6000 80FF C000
G C000
X C000 E0FF 5E00
W T 5E00 7EFF 5E00 TASMON
```

Finally, if you purchased the Model 4 version of TASMON but got version 1.0 of the INSTALL/CMD program, you will not be able to use TASMON4 with TRSDOS version 6.2. To fix this, enter the program below into an Editor-Assembler program and assemble it as INSTALL/CMD (or just use TASMON to enter the hex bytes into memory and write the result to disk as INSTALL/CMD). Then just use the new INSTALL/CMD program whenever you install TASMON4. Here's the instructions for using the new INSTALL program (which was revised to work with TRSDOS 6.2 by Mark S. Barlow):

INSTALL version 2.0 will now correctly install TASMON4 onto a disk formatted under either TRSDOS 6.1 or TRSDOS 6.2.

1) The files that come on your master TASMON4 disk are listed below with their function:

INSTALL/CMD - The program that will make SYS13/OBJ into a system file.

SYS13/OBJ - The overlay code that will become a system file.

TASMON4/CMD - This is the TASMON4 /CMD file - note that this file uses the SYS13/SYS file, which must be installed first!

TEST/CMD - A demonstration program.

2) INSTALL/CMD and SYS13/OBJ must be in ANY disk drive accessible by the system during the installation process (it is no longer a requirement that these files be on the disk in drive zero).

3) At the completion of the installation process, both INSTALL/CMD and SYS13/OBJ will be REMOVEd (killed) from the disk. Therefore, the proper procedure for installation is as follows:

a. COPY the following files from your TASMON4 master disk to any TRSDOS 6 system disk: TASMON4/CMD, INSTALL/CMD, and SYS13/OBJ. Then take your TASMON4 master disk out of the disk drive and put it away in a safe place.

b. Execute the INSTALL program (type INSTALL and press the ENTER key).

c. When the installation has been successfully completed, both TASMON4/CMD and a new SYS13/SYS file will remain on the disk.

It is important to note that in order to make copies of TASMON4, it will be necessary to either repeat the above procedure, or make a BACKUP of the entire system disk on which TASMON4 resides. TASMON4 will not work if you simply copy the TASMON4/CMD file over to another disk; it needs to have the SYS13/SYS file created by the INSTALL program present on the disk as well.

4) If INSTALL aborts back to TRSDOS with ANY sort of DOS error, it may be necessary to close some of the files that were opened during the unsuccessful installation process. This can be accomplished by using the RESET command in the form: RESET filename . Files that have been left open can be easily identified because they will have a question mark after their filename in the directory listing.

5) INSTALL will not work properly if the password protection of SYS13/SYS has been removed through the use of Super Utility Plus or a similar product. Either start again with a fresh backup of TRSDOS 6, or re-protect SYS13/SYS with the password "LSIDOS".

6) Possible error messages you may encounter during installation are as follows:

SYS13/SYS is occupied on your TRSDOS 6.1 disk, CANNOT install TASMON4. If you get this error message, TASMON4 has not been installed and is not functional. The only way this error could occur is if TASMON4 has already been installed on the disk or another program is using the SYS13/SYS slot (neither error is likely). In order to correct the problem, use a fresh backup of your TRSDOS 6 system disk.

An ECI is already installed, cannot install TASMON4. If you get this, TASMON4 was not installed because an ECI exists on that disk. ECI stands for Extended Command Interpreter, and this might be present if another applications program is using the SYS13/SYS file. To correct the problem, use another TRSDOS 6 system disk that doesn't have an ECI. Note that this error should rarely or never be encountered.

SYS13/SYS occupied on your TRSDOS 6.2 disk. This message simply indicates that the SYS13/SYS file was not an ECI, but did appear to contain some type of data (even if only "garbage" data). This will be the case when a fresh copy of TRSDOS 6.2 is used. Don't be alarmed; TASMON4 will be installed correctly and will be ready to use when the installation process is complete.

Here's the source code for the new INSTALL program:

```
                00001 ;INSTALL/CMD - The TASMON4/TRSDOS 6.x install program.
                00002 ;Written 07/19/84 by Paul F. Snively
                00003 ;Copyright (c) 1984 by The Alternate Source
                00004 ;MODIFIED BY MARK S. BARLOW 10/14/84 TO
                00005 ;HANDLE TRSDOS 6.2 AND MAKE MORE FRIENDLY
3000            00006          ORG     3000H           ;Right where we need it
3000 21B030     00007 START    LD      HL,OMESS        ;Point to opening message
3003 3E0A       00008          LD      A,10            ;SVC 10
3005 EF         00009          RST     28H             ;Do the SVC
3006 3E65       00010          LD      A,101
3008 EF         00011          RST     28H
3009 FD7E04     00012          LD      A,(IY+'E'-'A')  ;ECI? in SYS13/SYS?
300C B7         00013          OR      A
300D C29430     00014          JP      NZ,ECI
3010 112F32     00015          LD      DE,OBJFCB       ;Point to file control block
3013 219132     00016          LD      HL,BUFF         ;Point to buffer
3016 0600       00017          LD      B,0             ;LRL = 256
3018 3E3B       00018          LD      A,59            ;SVC 59
301A EF         00019          RST     28H             ;Try it out
301B 207F       00020          JR      NZ,ERROR        ;File's NOT there... error
301D 115832     00021          LD      DE,SYSFCB       ;Point to file control block
3020 219132     00022          LD      HL,BUFF         ;and to buffer
3023 0600       00023          LD      B,0             ;256 LRL
3025 3E3B       00024          LD      A,59            ;SVC 59
3027 EF         00025          RST     28H             ;Do SVC
3028 2072       00026          JR      NZ,ERROR        ;No such file
302A D5         00027          PUSH    DE              ;Move FCB from DE
302B DDE1       00028          POP     IX              ;To IX
302D DD7E0C     00029          LD      A,(IX+12)       ;Get first byte of ERN
3030 DDB60D     00030          OR      (IX+13)         ;Are there any records in file?
3033 2810       00031          JR      Z,COPY
3035 3E65       00032          LD      A,101           ;Get flag pointer
3037 EF         00033          RST     28H
3038 3E62       00034          LD      A,62H           ;IF TRS 6.2 then copy anyway
303A FDBE1B     00035          CP      (IY+27)         ;What version?
303D 2800       00036          JR      Z,COPY62
303F 3E3C       00037          LD      A,60            ;SVC 60
3041 EF         00038          RST     28H             ;Close file
3042 2058       00039          JR      NZ,ERROR        ;Go if error
3044 217331     00040          LD      HL,S13OCC       ;Else proclaim SYS13 occupied
3047 3E0A       00041          LD      A,10            ;Display it
3049 EF         00042          RST     28H             ;Go to!
304A 1845       00043          JR      EXIT            ;And exit
304C 3E0A       00044 COPY62   LD      A,10            ;Occupied and TRSDOS 6.2
304E 21EF31     00045          LD      HL,CPYANY
3051 EF         00046          RST     28H
3052 3E0A       00047 COPY     LD      A,10            ;Print "installing"
3054 211B32     00048          LD      HL,COMIT
3057 EF         00049          RST     28H
3058 CDA230     00050 COPYLP   CALL    GETREC          ;Get a record from source
305B FE1C       00051          CP      1CH             ;Done?
305D 280C       00052          JR      Z,CFILES        ;Yes, close files
305F FE1D       00053          CP      1DH             ;Done?
3061 2808       00054          JR      Z,CFILES        ;Yes, close files
3063 B7         00055          OR      A               ;Zero?
3064 2036       00056          JR      NZ,ERROR        ;Go if error
3066 CD0930     00057          CALL    PUTREC          ;Else write record
3069 18ED       00058          JR      COPYLP          ;and continue
                00059 ;
306B 115832     00060 CFILES   LD      DE,SYSFCB       ;CLOSE SYS13/SYS
306E 3E3C       00061          LD      A,60
```

```
3070 EF       00062          RST   28H
3071 2029     00063          JR    NZ,ERROR
3073 112F32   00064          LD    DE,OBJFCB       ;KILL SYS13/OBJ
3076 3E39     00065          LD    A,57
3078 EF       00066          RST   28H
3079 2021     00067          JR    NZ,ERROR
307B 117032   00068          LD    DE,INSTM        ;Open INSTALL/CMD
307E 219132   00069          LD    HL,BUFF
3081 3E3B     00070          LD    A,59
3083 EF       00071          RST   28H
3084 2016     00072          JR    NZ,ERROR
3086 3E39     00073          LD    A,57            ;Kill INSTALL/CMD
3088 EF       00074          RST   28H
3089 2011     00075          JR    NZ,ERROR
308B 212F31   00076          LD    HL,FINI         ;"Done" message
308E 3E0A     00077          LD    A,10            ;SVC 10
3090 EF       00078          RST   28H             ;Display it
3091 3E16     00079 EXIT     LD    A,22            ;Exit to TRSDOS
3093 EF       00080          RST   28H             ;Do it
3094 3E0A     00081 ECI      LD    A,10            ;Display "ECI" already there
3096 21BA31   00082          LD    HL,ECIM
3099 EF       00083          RST   28H
309A 18F5     00084          JR    EXIT
309C F640     00085 ERROR    OR    40H             ;Display brief
309E 4F       00086          LD    C,A             ;Move error to C
309F 3E1A     00087          LD    A,26            ;Display
30A1 EF       00088          RST   28H             ;Go to, boy!
30A2 112F32   00089 GETREC   LD    DE,OBJFCB       ;Point to object file
30A5 3E43     00090          LD    A,67            ;SVC 67
30A7 EF       00091          RST   28H             ;Do it
30A8 C9       00092          RET                   ;And return
30A9 115B32   00093 PUTREC   LD    DE,SYSFCB       ;Point to system file
30AC 3E4B     00094          LD    A,75            ;SVC 75
30AE EF       00095          RST   28H             ;Do it
30AF C9       00096          RET                   ;And return
30B0 1C       00097 OMESS    DEFB  28
30B1 1F       00098          DEFB  31
30B2 49       00099          DEFM  'INSTALL - TASMON4 installation program.'
    4E 53 54 41 4C 4C 20 2D 20 54 41 53 4D 4F 4E 34
    20 69 6E 73 74 61 6C 6C 61 74 69 6F 6E 20 70 72
    6F 67 72 61 6D 2E
30D9 0A       00100          DEFB  10
30DA 57       00101          DEFM  'Written by Paul F. Snively, Version 2.0'
    72 69 74 74 65 6E 20 62 79 20 50 61 75 6C 20 46
    2E 20 53 6E 69 76 65 6C 79 2C 20 56 65 72 73 69
    6F 6E 20 32 2E 30
3101 0A       00102          DEFB  10
3102 43       00103          DEFM  'Copyright (c) 1984 by The Alternate Source.'
    6F 70 79 72 69 67 68 74 20 28 63 29 20 31 39 38
    34 20 62 79 20 54 68 65 20 41 6C 74 65 72 6E 61
    74 65 20 53 6F 75 72 63 65 2E
312D 0A0D     00104          DEFW  0D0AH
312F 54       00105 FINI     DEFM  'TASMON4 has been installed.  Make BACKUPS from
    41 53 4D 4F 4E 34 20 68 61 73 20 62 65 65 6E 20          this working master.'
    69 6E 73 74 61 6C 6C 65 64 2E 20 20 4D 61 6B 65
    20 42 41 43 4B 55 50 53 20 66 72 6F 6D 20 74 68
    69 73 20 77 6F 72 6B 69 6E 67 20 6D 61 73 74 65
    72 2E
3172 0D       00106          DEFB  13
3173 53       00107 S130CC   DEFM  'SYS13/SYS is occupied on your TRSDOS 6.1 disk,
    59 53 31 33 2F 53 59 53 20 69 73 20 6F 63 63 75          CANNOT install TASMON4.'
    70 69 65 64 20 6F 6E 20 79 6F 75 72 20 54 52 53
    44 4F 53 20 36 2E 31 20 64 69 73 6B 2C 20 43 41
    4E 4E 4F 54 20 69 6E 73 74 61 6C 6C 20 54 41 53
    4D 4F 4E 34 2E
31B9 0D       00108          DEFB  13
31BA 41       00109 ECIM     DEFM  'An ECI is already installed, cannot install
    6E 20 45 43 49 20 69 73 20 61 6C 72 65 61 64 79          TASMON4.'
    20 69 6E 73 74 61 6C 6C 65 64 2C 20 63 61 6E 6E
    6F 74 20 69 6E 73 74 61 6C 6C 20 54 41 53 4D 4F
    4E 34 2E
31EE 0D       00110          DEFB  13
31EF 53       00111 CPYANY   DEFM  'SYS13/SYS occupied on your TRSDOS 6.2 disk.'
    59 53 31 33 2F 53 59 53 20 6F 63 63 75 70 69 65
    64 20 6F 6E 20 79 6F 75 72 20 54 52 53 44 4F 53
    20 36 2E 32 20 64 69 73 6B 2E
321A 0D       00112          DEFB  13
321B 49       00113 GOAIT    DEFM  'Installing TASMON4.'
    6E 73 74 61 6C 6C 69 6E 67 20 54 41 53 4D 4F 4E
    34 2E
322E 0D       00114          DEFB  13
322F 53       00115 OBJFCB   DEFM  'SYS13/OBJ'
    59 53 31 33 2F 4F 42 4A
3238 03       00116          DEFB  3
0017          00117          DEFS  23
3250 53       00118 SYSFCB   DEFM  'SYS13/SYS.LSIDOS:0'
    59 53 31 33 2F 53 59 53 2E 4C 53 49 44 4F 53 3A
    30
3262 03       00119          DEFB  3
000D          00120          DEFS  13
3270 49       00121 INSTM    DEFM  'INSTALL/CMD'
    4E 53 54 41 4C 4C 2F 43 4D 44
327B 03       00122          DEFB  3
0015          00123          DEFS  21
0100          00124 BUFF     DEFS  256
3000          00125          END   START
00000 TOTAL ERRORS

BUFF   3291   CFILES 3048   COPY   3052   COPY62 304C   COPYLP 3058
CPYANY 31EF   ECI    3094   ECIM   31BA   ERROR  309C   EXIT   3091
FINI   312F   GETREC 30A2   GOAIT  321B   INSTM  3270   OBJFCB 322F
OMESS  30B0   PUTREC 30A9   S130CC 3173   START  3000   SYSFCB 3250
```

ADDING EXTERNAL DISK DRIVES TO A MODEL 4P

I have had a few queries on how to add external disk drives to a TRS-80 Model 4P. I have been told that the Model 4P technical manual is now available (stock number 26-1080, $32.95) and it is not as complete as the Model 4 tech manual.

The process of adding external drives would not be very difficult at all for an experienced hardware hacker. Basically, the Model 4 has some circuitry that has been omitted in the 4P (another case of Radio Shack shooting themselves in the foot). It is actually possible to rewire the 4P to recognize all four drives without adding ANY additional components. The problem is that the Drive Select 2 and Drive Select 3 lines from U32 are left unconnected in the 4P. If you examine the circuit diagram for the 4P, the solution is fairly obvious. DS2 and DS3 (pins 7 and 10 of U32) have to be extended to pass through the driver (U20) and from there onto the drive cable bus. In addition, Drive Select lines 0 and 1 now feed into pins 2 and 3 of U75 (a NAND gate), this must be replaced with a four-input NAND gate with Drive Select lines 0 through 3 connected to the inputs. So how do you get away without adding components? Hint: U77 contains a five input NAND gate, of which only two of the inputs are currently being used. So if you partially switch the functions of U75 and U77... of course, you can always piggyback a new NAND gate IC on top of U75 and save yourself some trace cutting.

At this point in time, you have at least three choices for adding external drives to the 4P. You can get the tech manual and trace out the circuitry and "roll your own", or you can send your 4P to The Alternate Source and have a modification designed by their technician installed (sorry, it's not yet available in kit form - please call (517) 482-8270 or write to TAS for more information), or you can send $25 to United Software Computer Operations, 2522 Capital Circle N.E. Unit 16, Tallahassee, Florida 32308 (telephone (904) 386-5478) for a set of detailed, step-by-step instructions ("not for the beginner") for adding drives 2 & 3 to the 4P (these consist of two and a half pages of printed instructions, plus a schematic diagram of the modification). Just in case you live in or near Tallahassee, I'll mention that the folks at U.S.C.O. will also install their mod for an additional charge.

Thanks to Cy Galley for providing the information on the missing circuitry in the 4P. If anyone would like to work up detailed instructions for adding the additional drives that we could publish here in Northern Bytes, I'm sure many of our readers would appreciate it!

ANOTHER BUG IN THE MODEL I/III BASIC ROM

Try this line from BASIC READY:

PRINT 3# /2#

(The actual numbers used aren't important. The "#" specifies double precision). It bombs. Omit the space between the "3#" and the "/2#", and it works! The bug is in the ROM at 0EF2H. The byte found there is 23H, which is an INC HL instruction. If you could change it to a D7H byte, which is a RST 10H instruction, the bug would disappear! Thanks to Vern Hester (author of MULTIDOS and ZEUS) for pointing this one out to me!

PATCHES FOR ALE RUNNING ON THE LNW-80

These patches are for ALE (The Alternate Source's full screen Editor Assembler package) when used on the LNW-80 Model 2 computer. They enable the CONTROL key, and define the F2 function key as the ESCAPE key!

Memory location 5C5E: change CD 93 5B to AF C9 00
Memory location 5BBB: change E6 04 to E6 10
Memory location 5BD5: change 3A F3 57 E6 08 to 3A F7 57 E6 40
Memory location 5AED: change E6 07 to E6 17

VCLIST - a VisiCalc formula listing program
by Arne Rohde

The Basic VCLIST program will take a normal VisiCalc file in /VC form, and list any part of it to a printer, a disk file, or the screen. The formulae are listed in spreadsheet fashion, and may be listed with fixed or variable column width.

The program first prompts for the name of the /VC file containing the formulae to be listed. Then it asks for the starting and ending column and row names and numbers. Any rectangular block from the file may be listed. The program also asks whether variable or fixed column width is desired, and the actual or maximum column width.

All the data to be listed must be able to fit into the available string space. The data is read into memory, and if variable column width is required, the maximum width of any formula in each column is stored. A space is inserted between each column, and if an entry exceeds the maximum column width it will be noted in the listing.

When the data has been processed, the program will display the required printer width, and ask for the destination. This may be the screen display (D), the printer (P), a disk file (any filename except D, P, or Q) or the program may be terminated (Q). The quit option was added, since the program has been used in compiled form where the BREAK key will not terminate the program. The disk output could be used for processing with any word processor.

The program has been used on a Model I and a Model III, but it should also operate on a Model 4 with no more than minor modifications. It should be possible to write a similar program for listing information from a Multiplan file on the Model 4.

-Arne Rohde, R D 2, Whakatane, NEW ZEALAND

```
10 'VisiCalc display program, VCLIST/BAS
20 'Reads VC file and stores row/column formulas for printing
30 'Written by Arne Rohde, Pilevej 31, 7600 Struer, Denmark
35 'Now Arne Rohde, RD 2, Whakatane, New Zealand
40 'Version 1, July 1984
50 'File name: VCLIST/BAS
100 CLEAR 0: IF MEM>10000 THEN CLEAR MEM-5000 ELSE CLEAR
MEM/2
110 DEFINT A-Z: DEFSTR S-T: DEFDBL D
120 CLS
130 PRINT "VC List program - list VisiCalc formulas"
140 PRINT "Programmed by arne"
150 PRINT: PRINT "VC file containing formulas? ";
160 SDF="": LINE INPUT SDF
170 OPEN "I", 1, SDF
180 PRINT "Column name for first data? (A-BK)? ";
190 STY="": LINE INPUT STY: GOSUB 1000: TN=" A": LC=1
200 IF STY>"" THEN GOSUB 1100: IF STY="" THEN 180 ELSE
TN=STY: LC=IQ
210 PRINT "Row number for first data? (1-254)? ";
220 STY="": LINE INPUT STY: GOSUB 1000: RN=1
230 IF STY>"" THEN GOSUB 1200: RN=VAL(STY): IF RN<1 OR
RN>254 THEN 210
240 PRINT "Column name for last data? ("TN;"-BK)? ";
250 STY="": LINE INPUT STY: GOSUB 1000: TH="BK": HC=63
260 IF STY>"" THEN GOSUB 1100: IF STY<TN THEN 240 ELSE
TH=STY: HC=IQ
270 PRINT "Row number for last data? ("RN;"-254)? ";
280 STY="": LINE INPUT STY: GOSUB 1000: RH=254
290 IF STY>"" THEN GOSUB 1200: RH=VAL(STY): IF RH<RN OR
RH>254 THEN 270
300 PRINT "Fixed or variable column width? ";
310 STY="": LINE INPUT STY: GOSUB 1000: STY=LEFT$(STY,1): IF
STY="F" OR STY="V" THEN SFV=STY ELSE GOTO 300
320 IF SFV="V" THEN PRINT "Maximum c"; ELSE PRINT "C";
330 PRINT "olumn width (3-63)? ";
340 STY="": LINE INPUT STY: CW=10: IF STY>"" THEN GOSUB
1200: CW=VAL(STY): IF CW<3 OR CW>63 THEN 320
350 NC=HC-LC: NR=RH-RN: DIM T(NR, NC), W(NC)
360 FOR I=0 TO NC: IF SFV="F" THEN W(I)=CW ELSE W(I)=3
370 NEXT
380 LINE INPUT#1, S
390 IF EOF(1) THEN PRINT "End of VC input": GOTO 430
400 IF LEFT$(S,1)=">" THEN GOSUB 1400
410 PRINT S
420 GOTO 380
430 CLOSE
440 J=4:FOR I=0 TO NC: J=J+W(I)+1: NEXT
450 PRINT "Paper width required ="; J; "columns"
460 PRINT "Output file name or D, P, or Q? ";
465 'D=display, P=printer, Q=quit
470 STY="": LINE INPUT STY: GOSUB 1000
480 IF STY="Q" THEN 990
490 IF STY="" OR STY="D" THEN DP=1 ELSE IF STY="P" THEN
DP=-1 ELSE DP=0: OPEN "O", 1, STY
500 STY="  ": GOSUB 1800: FOR I=0 TO NC
510 STY=TN+STRING$(W(I)-1, " "): GOSUB 1800: STY=TN: GOSUB
1300: TN=STY
520 NEXT: GOSUB 1900
530 FOR J=0 TO NR: STY=STR$(RN): RN=RN+1: STY=RIGHT$(STY,
LEN(STY)-1): STY=STY+STRING$(4-LEN(STY), " "): GOSUB 1800
540 FOR I=0 TO NC: STY=T(J, I): T(J,I)="": IF LEFT$(STY,1)="/"
THEN SQ=RIGHT$(STY,1) ELSE SQ=" "
550 STY=STY+STRING$(W(I)-LEN(STY), SQ)+" ": GOSUB 1800
560 NEXT: GOSUB 1900
570 NEXT
990 CLOSE: END
999 'Convert STY to upper case
1000 IF STY="" THEN RETURN
1010 FOR IQ=1 TO LEN(STY): SQ=MID$(STY,IQ,1)
1020 IF SQ>="a" AND SQ<="z" THEN MID$(STY,IQ,1)=CHR$(ASC(SQ)-
32)
1030 NEXT: RETURN
1099 'Check valid column name and convert to numeric
1100 IF STY="" THEN RETURN ELSE IF LEN(STY)>2 THEN 1170
1110 IF LEN(STY)=1 THEN STY=" "+STY
1120 SQ=LEFT$(STY,1): IF SQ=" " OR (SQ)="A" AND SQ<="B") ELSE
1170
1130 IF SQ=" " THEN IQ=0 ELSE IF SQ="A" THEN IQ=26 ELSE IQ=52
1140 SQ=RIGHT$(STY,1): IF SQ<"A" OR SQ>"Z" THEN 1170
1150 IQ=IQ+ASC(SQ)-ASC("A")+1
1160 IF STY<="BK" THEN RETURN
1170 STY="": RETURN
1199 'check valid numeric input
1200 IF STY="" OR LEN(STY)>4 THEN STY="": RETURN ELSE IQ=1
1210 SQ=MID$(STY,IQ,1)
1220 IF SQ<"0" OR SQ>"9" THEN STY="": RETURN
1230 IQ=IQ+1: IF IQ<=LEN(STY) THEN 1210
1240 RETURN
1299 'Increment column name
1300 SQ=RIGHT$(STY,1): IF SQ<"Z" THEN
MID$(STY,2,1)=CHR$(ASC(SQ)+1): RETURN
1310 MID$(STY,2,1)="A": IF LEFT$(STY,1)=" " THEN
MID$(STY,1,1)="A" ELSE MID$(STY,1,1)="B"
1320 RETURN
1399 'Get formula, insert in array
1400 I=2: GOSUB 1500: GOSUB 1600: IF RV=0 THEN RETURN
1410 GOSUB 1100: CC=IQ-LC: CR=RV-RN
1420 GOSUB 1700
1430 T(CR, CC)=STY
1440 IF SFV="V" THEN IF LEN(STY)>W(CC) THEN W(CC)=LEN(STY)
1450 RETURN
1499 'get column and row numbers
1500 STY="": RV=0
1510 SQ=MID$(S,I,1): I=I+1
1520 IF SQ>="A" AND SQ<="Z" THEN STY=STY+SQ: GOTO 1510
1530 IF SQ>="0" AND SQ<="9" THEN RV=RV*10+ASC(SQ)-48: GOTO
1510
1540 I=I-1: RETURN
1599 'check column and row valid
1600 IF LEN(STY)=1 THEN STY=" "+STY
1610 IF STY<TN OR STY>TH THEN RV=0: RETURN
1620 IF RV<RN OR RV>RH THEN RV=0: RETURN
1630 RETURN
1699 'get formula, check for repeat entry
1700 STY=MID$(S,I,1): IF STY<>"|" THEN 1730 ELSE I=I+1
1710 IF MID$(S,I,1)<>"/" THEN 1730
1720 IF MID$(S,I+1,1)<>"-" THEN I=I+3
1730 STY=RIGHT$(S,LEN(S)-I+1)
1740 IF LEN(STY)>CW THEN STY=LEFT$(STY, CW-1)+">"
1750 RETURN
1799 'Print to screen, printer, or disk file
1800 IF DP>0 THEN PRINT STY; ELSE IF DP<0 THEN LPRINT STY;
ELSE PRINT #1, STY;
1810 RETURN
1899 'end line for screen, printer or disk
1900 IF DP>0 THEN PRINT ELSE IF DP<0 THEN LPRINT "" ELSE
PRINT #1, ""
1910 RETURN
```

HELP WANTED - CAN YOU HELP??

Mr. Dale Stupfel, 7563 Hillrose Drive, Dublin, California 94568 is looking for a program that will permit tapes produced by the Color Computer version of Scripsit to be read into the Model III (preferably into a Model III version of Scripsit or SuperScripsit). It should be possible, can anyone help?

TRSDOS 6.2 PREVIEW
By Dave Bower (MCI ID: 224-9092)

This is a preview of TRSDOS 6.2, not a review. If you are capable of running TRSDOS 6.2 then you have a Model 4/4P and already have TRSDOS 6.0 or 6.1 and you are familiar with the TRSDOS 6.x disk operating system. I will try to show some of the differences in 6.2 so you can decide whether or not you need/want to purchase this revision.

First, I make no claims to the accuracy of this article. All the information was gleaned from the TRSDOS 6.2 manual and the Disk System Owners Manual (TRSDOS 6.1) that comes with the Model 4/4P. And because the information is supplied solely by Radio Shack, I honestly doubt its accuracy. But it is close enough to serve our purpose.

TRSDOS 6.2.0 comes with the following:

- TRSDOS 6.2 diskette
- Complete Model 4/4P Disk System Owner's Manual
- TRSDOS 6.2 Hard Disk Initialization Diskette
- Hard Disk System Startup Manual (updated for 6.2)
- Revised pages for the Model 4 Technical Manual
- Software Section

Some of the new or different commands you pickup are:

CAT -- Displays a short directory (only the file names). Allows the same options as the DIR command. It is equivalent to the DIR (ALL=NO) command.

CLS -- Clears the screen via software. Useful in JCL's.

HELP -- This is by far the best help file I have seen in a DOS. It is extensive! There is actually a HELP program and a DOS HELP file. This leaves open the possibility for separate help files for different applications. And it can do a global search of all the help files on the diskette.

LOG -- Allows you to switch system diskettes without resetting the system. For example you could have a double sided (SIDES=2) diskette in drive 0 and want to switch to a single sided diskette. (This cannot be used to switch between versions 6.0, 6.1 or 6.2).

FLOPPY/DCT --This lets you define a logical drive as a floppy drive.

TOF -- Advances the printer to the top of the next page.

Some added parameters are:

DIR -- You can specify a range of drives. (DIR :0-:1)

FORMAT -- You can specify the directory track on a floppy format, you can recreate the directory file on a hard disk and you can specify SIDES=2 for a double headed floppy.

Double Sided Drive Support -- This may or may not be new, but it is undocumented. If you put double headed drives in your 4 or 4P, TRSDOS 6.2 will allow you to operate single volume. The command is FORMAT (SIDES=2).

SYSTEM -- A SMOOTH parameter supposedly disables the interrupts earlier during disk I/O and allows smoother disk access. But it messes up the type-ahead feature. SYSTEM (FAST & SLOW) is documented in the manual. The SYSTEM command now patches your system to run at 50 Hz.

CLICK/FLT -- Now has a CHAR parameter. This causes a CLICK everytime the specified character is encountered. The clicks are louder in 6.2. Now they are almost audible!

IEP -- TRSDOS 6.2 comes with an Immediate Execution Program. This is a program that is executed by pressing <*><ENTER> (asterisk, enter) from TRSDOS ready. The program that is executed is SYS13/SYS. If you rename your program, say SCRIPSIT for example, to SYS13/SYS.LSIDOS everytime you enter an asterisk from the keyboard, SCRIPSIT will be executed. The correct syntax would be:

COPY SCRIPSIT/CMD:1 SYS13/SYS.LSIDOS:0 (C=N)

You must have a SYS13/SYS file on the disk you are copying to.

As a little bonus you also receive compatibility problems. There is a warning that you MUST reset the computer before changing from a 6.1 to a 6.2 diskette. Otherwise your data may be damaged.

You receive an instruction sheet explaining how to upgrade from 6.1 to 6.2. It even recommends updating your data diskettes. You cannot upgrade directly from 6.0 to 6.2. You must first upgrade from 6.0 to 6.1 and then to 6.2. (I never saw 6.0, but it must have been a winner. An addendum in the 6.1 manual says that 6.0 is not compatible with the 4P.) But then you buy a computer from a toy store, what do you expect.

TRSDOS 6.2 appears to boot and run faster than 6.1. But given the experience that Radio Shack and Logical Systems Inc. (TRSDOS 6 is supposed to be LDOS) have in this field, it should NOT have taken 3 revisions to achieve this! In fact there are no real revelations in TRSDOS 6.2. Most of these features have been around for a few years, on Radio Shack machines.

Summary stuff:

In spite of what I said, I bought it. The major irritations (for me, at least) are in the way that Radio Shack has handled distribution and revision of TRSDOS 6. The DOS works, and works well! The help file alone is worth the price. This version should be issued with the computer, because of the help file. If you are going to stick with TRSDOS you may want to "keep current" since revisions have not been compatible with one another. So far it is a very good DOS at a very good price. But, if each revision costs as much as 6.2, it will soon pass the cost of the higher priced brands.

One more item, it is available. Not all Radio Shack stores know this or know what it is. Don't take their word for it, look on the shelf. It is labelled "TRSDOS VERSION 6" on the binder and should have a 06.02.00 sticker somewhere. That's good advice for anything you're looking for, even in the computer center. I was told by a salesman in the computer center that the Model 4 Technical Reference Manual was not available through stores and had to be special ordered. I found it on the shelf.

THE OTHER TYDBYTE
by Robert Filipiak

[Reprinted from ELSE newsletter.]

This tydbyte was forced out of the last issue because of space limitations. Have you ever wanted to determine the number of days between two dates?? Not the easiest thing to do now is it?? Have you ever wanted to compress an 8 byte date string into something smaller??? I have seen this subject covered only once in one of our trade newsletters. I spotted that attempt in the Valley Forge newsletter of November, December 1982 and January 1983. There were some limitations to its use however, so I thought this one might be of much more value to our readers. The concept was filched (WHAT??!!) from a CP/M program so I don't take all of the credit.

This routine below takes an 8 byte string in the form "mm/dd/yy" and converts it into 2 bytes. (YES that's TWO bytes.) I am leaving the check for validity on the string up to you. I will explain how the variables are used below the routine. Spaces have been left in the program lines for readability.

Squeeze a "mm/dd/yy" date into a 2 byte integer:

```
110 MO%= VAL(LEFT$(CD$,2)): DA%= VAL(MID$(CD$,4,2)):YR%=
VAL(RIGHT$(CD$,2)): IF (MO%<0) OR (MO%>12) OR (DA%<0) OR
(DA%>31) THEN PRINT" INVALID FORMAT":CD%=32767:RETURN
111 CD%=INT(YR% * 365.25 + 0.75)+ MO%(MO%) + DA% - 18263: IF
((YR%/4)-INT(YR%/4)<>0) AND MO%>2 THEN CD%= CD%-1:RETURN
ELSE RETURN
```

One note to the above, an integer array MO%(N) for elements 1 through 12 inclusive must have been DIMensioned before using this subroutine. The values for that array are:

```
19 DIM MO%(12):MO%(1)=-1: MO%(2)=30: MO%(3)=59: MO%(4)=90:
MO%(5)=120: MO%(6)=151: MO%(7)=181: MO%(8)=212: MO%(9)=243:
MO%(10)=273: MO%(11)=304: MO%=334
```

How does it work?? It is easy; you enter the routine with CD$ containing the 8 byte date string. MO%, DA% and YR% are the integer values for the component parts of the date string. The number of years (YR%) is multiplied by 365.25 (remember LEAP years) to obtain the number of days that has elapsed from the beginning of the CENTURY to the beginning of the year. Then the number of elapsed days for the current month (MO%(MO%)) and into the month are added to this figure. It is adjusted for leap years with the "IF" statement in line number 111. Then 18263 is subtracted from the total to give you a number in CD%. You probably wonder why do we subtract 18263???? I did for a couple of days and then tried it out to find out why.

A little bit of simple arithmetic is needed (use your computer, what else do you have it for?). Without even allowing for leap years, if you start at the beginning of the century on 01/01/00 (remember 1900) and count the number of elapsed days to 12/31/99; you will quickly realize that there are over 36,500 days in a century. That is too big for an integer; so why don't we base this conversion at the exact middle of the century (01/01/50) and any date in the century will be either positive (01/01/50 or later) or negative (before 01/01/50). To set the correct number of elapsed days from the beginning of the century, just add 18263. (There are 18263 days

from 01/01/00 to 01/01/50.) This value is used as an "offset" in order to allow the "converted" date to be in the range of -32768 to +32767.

Aside from the obvious uses in compacting record space in disk files; what else can we use this for? If you have any application requiring the number of days between two dates; or need to know what date is 35 days from today, this conversion routine makes those chores easier.

OK, now that we have crunched the date into an integer, I am sure you have the need to stretch it back into an 8 byte string. Well, this routine will accomplish that.

Unsqueeze a two byte integer into an 8 byte string:

```
120 YR%= INT((CD%+18263)/365.25): DA%= INT((18263+CD%)-
YR%*365.25): MO%=12: IF ((YR%/4)-INT(YR%/4)<>0) AND DA%>58
THEN DA%=DA%+1
121 IF DA%<=MO%(MO%) THEN MO%=MO%-1: GOTO 121 ELSE
DA%=DA%-MO%(MO%)
122 CD$= RIGHT$(STR$(MO%),2) +"/"+ RIGHT$(STR$(DA%),2) +"/"+
RIGHT$(STR$(YR%),2): RETURN
```

This routine works by taking the converted date (CD%) and adding our offset to it to obtain the number of elapsed days to the beginning of the year. Then the number of elapsed days for this year are determined. The correct month is found by continually checking thee number of days in the year until they fall below the number in the year. The subscript of the array is our month number. Some string conversions and assembly; the result is CD$ or our 8 byte date string. Try it sometime.

NEW COMMUNICATIONS PROTOCOL

The September, 1984 issue of Link-Up magazine featured an article about MEX, a new communications program developed by Ron Fowler of Fort Atkinson, Wisconsin. MEX is an acronym for "Modem Executive" and started out as an adaptation of Modem 7, Ward Christensen's popular public domain communications program that is used on many CP/M based systems. By the time Ron got through, however, he had totally rewritten and revised the program. Mr. Fowler operates his own bulletin board (The "Fort Fone File Folder") at (414) 563-9932 (it is an RCP/M board, and if you are not familiar with this type of board you may have some difficulty getting around on it), and he also has accounts on Arpanet, CompuServe, and MCI Mail (his MCI Mail ID is 210-3334). His mailing address is: Route 1 Box 7, Fort Atkinson, Wisconsin 53538. Since it is entirely possible that someone may want to try and adapt this program for the TRS-80, I decided to ask Mr. Fowler some questions, and received the following replies:

Q. What is the MEX protocol? How is it similar to Ward Christensen's original MODEM protocol? How is it different? How compatible are the two protocols?

A. MEX currently handles two protocols, and sort of a third: Christensen (fully compatible with the earliest versions, to the best of my knowledge), CompuServe 'A' protocol, and an extension of the Christensen protocol (often called 'batch-Christensen') that allows many files to be transferred sequentially, without operator intervention. MEX also supports the original Checksum method of error-detection, as well as the newer CRC versions.

Q. What versions of MEX are currently available (for which computers? Which Models?). Specifically, are any versions available (to your knowledge) for the TRS-80 Models I or III (or 4 or 4P running in the Model III emulation mode)? If so, how can these be obtained?

A. This is a hard one even for me to answer. My original release ran only on the PMMI modem (a now-obscure S100 bus board). Shortly thereafter, I released an overlay for a Godbout serial card. Within a few weeks, users were writing overlays faster than I could keep track of -- I believe there are overlays now for about 40 or so computers, including the following TRS80 models/versions:

MXO-R+10.ASM: MOD IV, CP/M+
MXO-R211.ASM: MOD II/12/16
MXO-RS13.ASM: MOD IV, Montezuma Micro CP/M 2.2

There may be other TRS overlays that I'm not aware of, but these are the ones currently appearing on our board. In addition, there are overlays available for such computers as Kaypro, Osborne, Osborne Executive, Epson, Morrow, Northstar, Xerox, Sanyo, Televideo, Heath and many others. Also "logical" modem overlays (i.e., those that interface MEX to intelligent modems by sending command strings through the "physical" overlay) for the Racal-Vadic, Hayes (including lookalikes such as US Robotics, Anchor, Promethius, CTS) and a few others.

Q. Can you provide enough information about MEX so that a competent programmer could write a version of MEX for a system not currently supported? This would be of special interest if there is not currently a TRS-80 version available.

A. I've always hoped to write an Installation Guide, but haven't so far been able to find the time. Most of the overlays were developed by emulating other overlays, dating back to the first PMMI overlay (which was, and is, the most well documented, since it was meant to serve as a model upon which others could be written -- it's current revision name is MXO-PM21.ASM).

Other comments by Mr. Fowler:

"A good deal of information is covered in the MEX user's manual, which is distributed with the basic package. It's about 80K long, [and is] available on our local BBS (Fortfone, mentioned in the article); since it's part an LBR file, you'll need to use the L option of our local XMODEM to receive just the documentation file. In addition, you'll need a local unsqueeze utility. (MEX112.LBR is the current revision's LBR, member name is MX110DOC.WQ, and the entire package including all user-written overlays is available in user 0, drive B on FortFone).....

"There are a number of newsletters that were written during the summer to provide bug fixes and tips for users. With the advent of MEX 1.1, however, the bug fixes no longer apply, and most of the tips have been incorporated into the manual as appendices.

"I should mention that I'm currently developing a much-enhanced version (MEX 2.0) that includes, among other things, a built-in language interpreter that will allow user-written MEX "programs" to completely replace the user interface (it includes such things as integer and string variables, long variable names, an expression analyzer that includes integer and string comparisons as well as access to STAT variables, and IF/GOTO constructs). The new version will also support I/O redirection, LBR files and SQueezed files, as well as disk-based commands that are loaded into memory only as needed (there will be a Programmer's Guide to allow users to write their own non-resident assembly-code commands). MEX2 will be a commercial effort, and will retail for somewhere between $80 and $100. Over a period of time, we'll be adding support for as many computers as possible, including several models of TRS80, as well as a native-code version for the IBM-PC.

"We're building a mailing list of people who'd like to be notified when MEX2 is available. Anyone who'd like to leave their name and address (and equipment type, if possible) may do so either by leaving a private message on FortFone, or sending a postcard to the above address."

One final note - Mr. Fowler currently only has the capability to write 8" SSSD diskettes, so don't send him a blank TRS-80 disk and ask him to make you a copy, because he can't do it. If you live in an area with an RCP/M Bulletin Board System, you may be able to download a copy of MEX from that board instead of calling Wisconsin. Since MEX is in the public domain, it could form the basis of a good TRS-80 (non-CP/M) communications package. If anyone knows of anything like this, please pass the information along to us!

RSCOBOL PATCHES
by Jim Whittaker

[Reprinted from the Sydney (New South Wales, Australia) TRS-80 Users Group Newsletter.]

These patches are to allow owners of the RSCOBOL system (supplied by Radio Shack) to transfer to other DOS's.

The original COBOL, both compiler and runtime, are the same for MODEL I and MODEL III. To do this the program checks the machine it is running on (by peeking into ROM and seeing if the character after "MODEL I" is a "I") and then does the appropriate setups. It then checks a signature byte in the DOS system RAM (4000H to 4200H) to ensure it is running on TRSDOS presumably to ensure correct file operation. It is here that you get runtime and compiler errors when you use COBOL with different DOS's.

The programs both do a compare with what it expects to find and jumps to an error routine depending on the flags that are set. The quickest way to disable this is to disable the compare. The assembly instruction CP (HL) is converted into the number BE (hex) so we must change this to 00 (hex) which is a NOP or no operation.

RSCOBOL:
Memory location 9A4BH change BE to 00.
This is at relative sector 5A, byte 5F in the disk file.
RUNCOBOL:
Memory location 9B37H change BE to 00.
This is at relative sector 5B, byte 87 in the disk file.

You can either patch the file or zap the disk to install it for good. NOTE: There are some other patches published for some of the DOS's to allow correct file manipulation under COBOL.

UPGRADING THE TRS-80 MODEL I TO 64K OF DRAMs

by Errol Rosser

[This article is reprinted from the Sydney (New South Wales, Australia) TRS-80 Users Group Newsletter. North American readers might question the use of the word "track" within the article, we use the word "trace" to mean the same thing (the conductive foil circuit paths on a printed circuit board), so just mentally substitute "trace" where you see "track". Just in case you need it, Mr. Rosser's phone number is (02) 709-7646 (from the U.S. and Canada, 011+61+2+709-7646).]

For those readers who wish to convert their Model I's to 4164 type 64k DRAMs (Dynamic Random Access Memory), here is a modification to allow uses of Motorola 6665's or similar 128 cycle refresh 4164's. The T.I. TMS4164 is unsuitable for this application because it requires a 256 refresh cycles and the Z80 only generates 128 refresh cycles then repeats the same 128 cycles afterwards. This happens because the Z80's refresh register (R) is a 7 bit binary counter with the eighth bit as a separate single bit memory changed only by the software and the "Master Reset" signal.

These alterations have been successfully made on both a 'D' series and a 'G' series board and the track layouts were the same. If anyone finds a Model I that doesn't match, please contact me.

Please note that these modifications require the electronics board to be removed from the case and much care should be used when cutting the tracks and when soldering the straps. I used an art knife with a small blade to cut the tracks, a magnifier to check that the cuts were clean, and wire-wrap wire for the straps. The whole job took approximately two hours to complete, including testing so don't rush in expecting to do it in 10 minutes.

Figures 1, 3, & 4 are on the circuit side and 2 is on the component side.

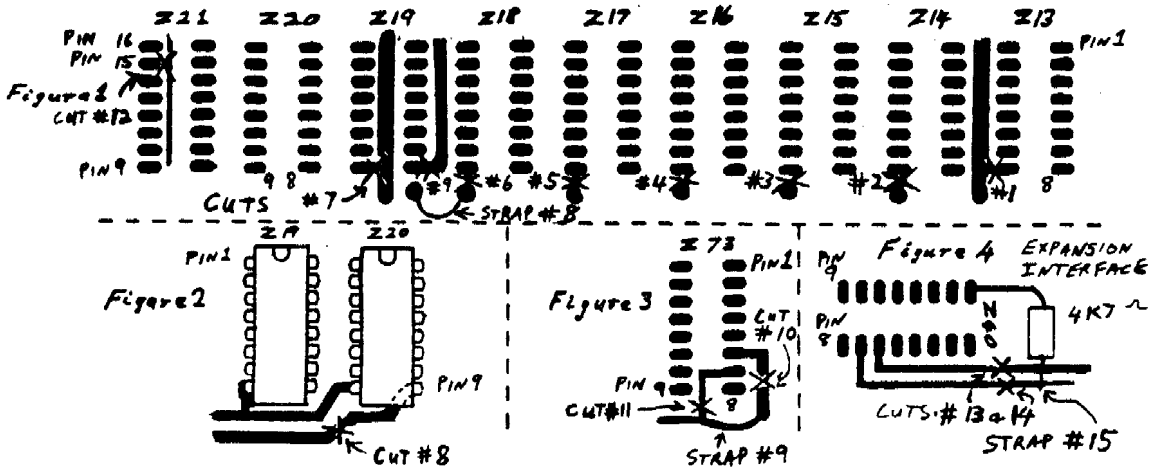
Part 1 Tracks to be cut:

(a)	Remove +5V from pin 9 of Z13 to Z20	cuts #1-#8	figures 1 & 2
(b)	Remove +12V from pin 8 of Z19	cut #9	figure 1
(c)	O/C *RAS from Z73 pin 5	cut #10	figure 3
(d)	O/C Z73 pin 6 from Z21 pins 2 & 14	cut #11	figure 3
(e)	O/C A14 from Z21 pins 1 & 15	cut #12	figure 1

Part 2 Straps to be added:

(a)	+5V to Z19 pin 8	strap #8	figure 1
(b)	Z13 - Z20 pin 9 to Z51 pin 12	straps #1 - #7	
(c)	*RAS to pins 2 & 14 at Z73	strap #9	figure 3
(d)	Z73 pin 6 to Z21 pins 1 & 15	strap #11	
(e)	Z38 pin 11 to Z73 pin 5 & Z51 pin 14	straps #11 & #12	
(f)	Z38 pin 9 to Z51 pin 13	strap #14	

If you have an expansion interface then one alteration will be required in it to prevent the selection of the RAM sockets in there. The tracks to Z40 pins 6 & 7 are to be cut and then joined together and pulled up to +5V by a 4K7 resistor as per cuts #13 & #14 and strap #15 in figure 4.

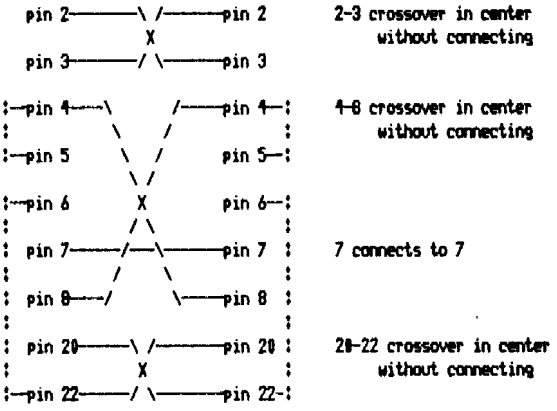


A NULL MODEM by Jeff Lasman

[Reprinted from the Valley Computer Club Newsletter.]

Sometimes it's easier to hook up two computers through modems than it is to connect them when they're sitting next to each other. The modem connection is pretty standard by now and the cable shown below should work in place of a modem, with only one CAVEAT.

Only the wires shown should be connected, providing that both computers normally use "straight-thru" cables (i.e. pins two and three NOT crossed). Set both computers to the same baud rate, and transfer away!



Lines 4 and 5 are shorted together on each end of the cable, but are not connected through the cable. Pins 6 and 22 are handled likewise. Good Luck!

SOME HELPFUL STUFF?

by Dave Bower (MCI ID: 224-9092)

Did you ever try to find something when you really needed it? Sure you have. Were you ever able to find something when you tried to find it when you really needed it. Sure you weren't. This is especially true of tidbits of information that really have no real place to go. I mean do you file tidbits of information under T for tidbit, or I for information?

I find the best place to store little hints, tricks, pokes, peeks and assorted tidbits that may become important some day, is in a newsletter. Newsletters (such as the one you are holding) contain a wealth of information, are solid information, a lot of undocumented information and are concise, easy to store and easy to search through. Especially if you put a little note on the front page about what's of interest to you inside. Also, newsletters are printed in small print and 2 columns. You can photocopy your hint, cut it out and tape it by your computer, if appropriate. And YOU don't have to bother typing it in and formatting the print-out yourself. So, with Jack's permission, I would like to store some things away in Northern Bytes for future reference. There's nothing newsbreaking here. You could find most of these in the manuals yourself. But it took me a while to find it (I couldn't find it when I needed it) and so now that I have it together I need someplace to store it so I can find it next time I need it. And a couple things I'm going to cut out and tape near my machine.

I'm going to talk about speeding up the Model 4 when in the Model III mode. I'd seen it a hundred times, but it took me a few days to come up with it when I really needed it! I'm also going to provide a list (and brief explanation) of the SYSTEM files and special FILTER and utility files on TRSDOS 6. That's nice to know if you want to make a minimum system disk. Or, if you REALLY need an extra 1.5K on a disk. And some you may never-ever need, so why keep them around on your working disk. (I'll give you the

passwords for TRSDOS 6 files, too). I'll list a DO file for TRSDOS 6 that will allow you to do a BACKUP BY CLASS on two data disks. And lastly (is lastly a real word?) I've got a WARNING/HINT for people that have occasion to use single-volume and double-volume drives (especially with DOSPLUS). Say for example someone that has a Model III with double headed drives and then has access to a Model 4/4P at the office.

SPEED UP

Now for the Model 4 speed-up in the Model III mode. I never did find this in any manual, this is courtesy of friends.

```
1 PV = PEEK(16912) 'Get present value
2 NV = PV OR 64    'Set bit 6 (64 = 40hex, 0100 0000 binary)
3 POKE 16912, NV   'Save new value
4 OUT 236, NV      'Set port
```

To reset the speed change line 2 to "2 NV = PV AND 191".

How about two programs to do it from DOS. Ok, I call these programs FAST4/CMD and SLOW4/CMD.

FAST4/CMD

```
         ORG    5700H
START    LD     A,(4210H)
         SET    06H,A
         LD     (4210H),A
         OUT    (0ECH),A
         JP     402DH
         END    START
```

SLOW4/CMD

```
         ORG    5700H
START    LD     A,(4210H)
         RES    06H,A
         LD     (4210H),A
         OUT    (0ECH),A
         JP     402DH
         END    START
```

SYSTEM FILES

You don't need a full set of system files (/SYS) on a TRSDOS 6 disk to operate. Here is a list of what they are and what they do. (The password for TRSDOS 6 system files is LSIDOS).

NOTE: DIR/SYS and BOOT/SYS should never be removed or copied from one disk to another.

SYS0/SYS Contains the resident part of the operating system. Used to allocate file space. Must be on a disk used for booting the system.
SYS1/SYS Must be on all SYSTEM disks.
SYS2/SYS Must be on all SYSTEM disks.
SYS3/SYS Must be on all SYSTEM disks.
SYS4/SYS Contains system error messages. It is recommended that you do NOT remove this file. But if you do, any system error will result in a SYS ERROR message.
SYS5/SYS TRSDOS DEBUG. If you do not intend to load DEBUG from this disk then remove it.
SYS6/SYS Contains Library A section of the LIB command. There's not much you can do without this one.
SYS7/SYS Contains Library B section of the LIB command. You can get away without this one.
SYS8/SYS Contains Library C section of the LIB command. Keep it if you need it.
SYS9/SYS More DEBUG stuff. Remove it if you do not intend to use DEBUG. If you removed SYS5/SYS then this is useless.
SYS10/SYS Needed to REMOVE a file. Don't remove it.
SYS11/SYS Needed to perform Job Control Language stuff. Cannot work without SYS6/SYS (Library A, which contains the DO command).
SYS12/SYS Must be on all SYSTEM disks.
SYS13/SYS On version PRIOR to TRSDOS 6.2 this was reserved for future use and serves no purpose, OFF WITH ITS HEAD! But on TRSDOS 6.2 it is only needed if you intend to use the Extended Command Interpreter (if you don't know what it is, you don't need it) or an Immediate Execution Program. Otherwise kill (excuse me, remove) it.

UTILITY PROGRAMS (that you may not need)

COMM	Communications package
CONV	Copy from Model III TRSDOS to TRSDOS 6
DOS/HLP	Help file for TRSDOS 6.2
HELP/CMD	TRSDOS 6.2, provides online help
LOG	Used to switch system diskettes without rebooting
PATCH	Used to change existing files
REPAIR	Used to correct(?) non-TRSDOS diskettes
TAPE100	Tape/Disk utility for Model 100

DEVICE DRIVERS

COMM/DVR	RS-232 driver
FLOPPY/DCT	Not needed in a floppy-only system
JL/DVR	The Joblog driver
MEMDISK/DCT	Allows you to add a "drive" in memory

FITLERS

CLICK/FLT	Produces key clicks
FORMS/FLT	Printer parameters
KSM/FLR	Assigns your commands to individual keys

PASSWORDS, so that you can remove the above mentioned files.

System files	LSIDOS
Filter files	FILTER
Driver files	DRIVER
Utility files	UTILITY
BASIC (& OVERLAYS)	BASIC
CONFIG/SYS	CCC
/DCT files	UTILITY

BACKUP BY CLASS

The BACKUP BY CLASS option on TRSDOS 6 is handy, sort of. I would normally work with a data disk in drive 1, and I would want to backup the files I modified during the day. But a Model 4P (and most Model 4's) have only 2 drives, so what good is it? At the time I needed it, it was no good because I didn't have the manuals at the office. But, you can put the necessary SYSTEM files into memory with the SYSRES command. Now the manual says you only need SYS files 2, 3 and 10, but to backup MODified files on a data disk by class you also need SYS12/SYS. I call the following file BYCLASS/JCL and when I'm working at the office I enter the command "DO BYCLASS/JCL" before I start work. Then forget about it. To bacop ryour data disks use the command "BACKUP :0 :1 (MOD,X)". TRSDOS will prompt you for the disk mount.

```
. This will put SYS files 2, 3, 10  and 12 into memory
.   for a Backup-by-class with no System disk in drive 0
.
SYSTEM (SYSRES=2)
SYSTEM (SYSRES=3)
SYSTEM (SYSRES=10)
SYSTEM (SYSRES=12)
.
. October 9, 1984
.
```

Lastly (it's not a word, I checked) you DOSPLUS people beware. Firstly, you'll notice that you can read TRSDOS 6 diskettes (as data disks) and TRSDOS 6 can read your DOSPLUS diskettes. That is because the Model 4/4P comes with SINGLE SIDED disk drives. And TRSDOS 6 (which is suppose to really be LDOS) is compatible with DOSPLUS when using single sided drives. But when you use single volume drives with TRSDOS 6 it puts the directory on BOTH sides of the disk, using the whole cylinder. DOSPLUS uses a short directory, only using one side of the track. They are not compatible!

Also be VERY careful when swapping diskettes between a Model 4/4P with single headed drives and your double headed Model III with DOSPLUS. I work on a Model 4 at work and bring it home to print on my Model III. I use DOSPLUS at home and it is configured to AUTO boot my applications programs and it assumes that all diskettes are single volume. And to write to a diskette with DOSPLUS set with the SIDES parameter wrong is sudden death. I've written a little DO file to prompt me to make sure all diskettes are mounted and then do a "I,M" to check all the drives, and then it loads the program.

That's all folks.

HELP WANTED - CAN YOU HELP??

Mr. David Librik, 2211 East Reservoir, Peoria, Illinois 61614 would like to buy a previously-owned copy of the disk version of Microsoft's Editor/Assembler-Plus for the TRS-80. The original documentation must be included!

UNDOCUMENTED Z-80 OPCODES
by Bill Smythe

[Once upon a time there was a group called the Chicago TRS-80 User Group, and they had a publication called CHICATRUG NEWS, and it was one of the best user group newsletters around. Unfortunately, the group has since disbanded, and CHICATRUG NEWS is no more, and TRS-80 users are poorer for that. The article which follows is reprinted from CHICATRUG NEWS, and has previously appeared in NORTHERN BYTES Volume 3, Number 10. I have had several occasions to refer back to this article since then, so I think maybe it deserves another exposure, and I hope you agree. By the way, if anyone can provide me with the current address of Bill Smythe, I would very much appreciate it!]

If you are willing to set aside your Editor/Assembler for a while and take a direct look at the Z-80 opcodes, certain patterns will become apparent. For example, of the 256 possible opcodes (00 through FF) all but four are in use as one-byte instructions. Some of these stand alone; others are followed by one- or two-byte operands. For example:

```
13          INC DE          (no operand)
3E nn       LD A,nn         (1-byte operand)
10 dd       DJNZ dd         (1-byte operand)
01 nn mm    LD BC,mmnn      (2-byte operand)
```

The four missing opcodes are CB, DD, ED, and FD. These serve as "escape codes" to alert the Z-80 that a two-byte instruction is coming up. Examples:

```
CB C0       SET 0,B
DD 21 nn mm LD IX,mmnn
ED B0       LDIR
FD 2B       DEC IY
```

Like the one-byte instructions, some two-byters are used with operands, while others stand alone.

One of the first observations made by any Z-80 hacker is the parallel among HL, IX, and IY instructions:

```
   21 nn mm LD HL,mmnn
DD 21 nn mm LD IX,mmnn
FD 21 nn mm LD IY,mmnn

   77       LD (HL),A
DD 77 jj    LD (IX+jj),A
FD 77 jj    LD (IY+jj),A
```

Most any instruction referring to HL can be changed to the corresponding instruction for IX or IY by prefixing DD or FD, respectively. When HL appears in parentheses, it becomes (IX+jj) or (IY+jj), where the index jj appears in the opcode as the second byte following the DD or FD. Meanwhile, the two-byte opcodes beginning with CB form an orderly set of shift, rotate, set, reset, and bit-test instructions, while the relatively disorganized ED set performs a variety of useful tasks.

And then there are the combined prefixes DDCB and FDCB. Their relationship to the CB codes are pretty much what you would expect:

```
   CB 46    BIT 0,(HL)
DD CB jj 46 BIT 0,(IX+jj)
FD CB jj 46 BIT 0,(IY+jj)
```

THE FUN BEGINS

Everything said so far is pretty much common knowledge, well-documented by Zilog, the Z-80 manufacturer. But what happens when DD or FD is prefixed to an instruction not containing HL?

```
   37          SCF
   C3 nn mm    JP mmnn

DD 37          ?????
FD C3 nn mm    ?????
```

As far as I can determine, the prefixes in these cases have no effect. The two instructions shown here continue to function as SCF and JP mmnn, respectively.

But what if the instruction deals with either H or L, but not HL?

```
   24          INC H
   2E nn       LD L,nn
```

Just as the register pair HL is made up of two registers, H and L, it appears that the 16-bit IX register consists of two 8-bit registers, which I shall call HX and LX. Similarly for IY. The above instructions become:

```
DD 24          INC HX
DD 2E nn       LD LX,nn
FD 24          INC HY
FD 2E nn       LD LY,nn
```

The last instruction, for example, loads the low-order half of the IY register with the value nn while leaving the high-order half untouched.

If, however, an instruction contains both H and (HL), or both L and (HL), then only the (HL) part is affected by the addition of DD or FD:

```
   66          LD H,(HL)
DD 66 jj       LD H,(IX+jj)
FD 66 jj       LD H,(IY+jj)
```

Adding a second DD or FD in front has no additional effect. Apparently, there are no such instructions as LD HX,(IX+jj) or LD HY,(IY+jj).

WHAT'S YOUR CB HANDLE?

The CB instructions are divided into four groups:

```
CB00 through CB3F:   rotate and shift group
CB40 through CB7F:   BIT testing
CB80 through CBBF:   RESet group
CBC0 through CBFF:   SET group
```

Of these four groups, all seem complete except for the first. The first group is divided into eight subgroups of eight instructions each:

```
CB00-CB07:  RLC (rotate left circular)
CB08-CB0F:  RRC (rotate right circular)
CB10-CB17:  RL (rotate left through carry)
CB18-CB1F:  RR (rotate right through carry)
CB20-CB27:  SLA (shift left arithmetic)
CB28-CB2F:  SRA (shift right arithmetic)
CB30-CB37:  ???????
CB38-CB3F:  SRL (shift right logical)
```

The missing CB30 group looks as though it ought to be a shift left logical, whatever that is. Trouble is, SLA (shift left arithmetic) is already pretty logical. So what's left for SLL to do? As might be expected, SLL shifts bits 0 though 6 leftward into bits 1 though 7, while bit 7 goes into the carry flag. But then comes the surprise -- bit 0 is set. Yes, Virginia, regardless of the previous status of any bit, or of any flag, bit 0 is turned on. Thus SLL, in effect, multiplies by 2 and adds 1. As with the other CB instructions, the affected register is B, C, D, E, H, L, (HL), or A depending on which instruction, CB30 through CB37 is used.

COMBINATIONS AND MORE COMBINATIONS

Of the combined opcodes, DDCBjjxx and FDCBjjxx, only every eighth one is documented by Zilog:

```
DD CB jj 06    RLC (IX+jj)
DD CB jj 0E    RRC (IX+jj)
     (etc.)
```

One might not expect much from those not on the list, since the corresponding DD-less CB instructions have nothing to do with HL. Not so, however -- a lot of weird stuff is going on here:

```
DD CB jj 00    RLC B,(IX+jj)
DD CB jj 01    RLC C,(IX+jj)
DD CB jj 02    RLC D,(IX+jj)
DD CB jj 03    RLC E,(IX+jj)
DD CB jj 04    RLC H,(IX+jj)
DD CB jj 05    RLC L,(IX+jj)
DD CB jj 06    <documented as above>
DD CB jj 07    RLC A,(IX+jj)
```

-- and similarly for RRC, RL, RR, SLA, SRA, SLL, and SRL. But what does that mean, "RLC B,(IX+jj)"? That's just a name I chose for a peculiar phenomenon in which (IX+jj) is rotated left circular, then copied into B. In other words, RLC B,(IX+jj) is like RLC (IX+jj) followed by LD B,(IX+jj).

The SET and RESet instructions are even more curious:

```
DD CB jj 80    RES B,0,(IX+jj)
DD CB jj 81    RES C,0,(IX+jj)
DD CB jj 82    RES D,0,(IX+jj)
DD CB jj 83    RES E,0,(IX+jj)
DD CB jj 84    RES H,0,(IX+jj)
DD CB jj 85    RES L,0,(IX+JJ)
DD CB jj 86    <documented: RES 0,(IX+jj)>
DD CB jj 87    RES A,0,(IX+jj)
```

I don't know how many Editor/Assemblers could handle 3 operands, even if they recognized undocumented opcodes. I couldn't think of any other way to express what happens -- (IX+jj) first has bit 0 reset, then is copied into the indicated register (e.g. B). The action is equivalent to RES 0,(IX+jj) followed by LD B,(IX+jj). Of course, the same can be done with bits 1 through 7, with SET as well as RESet, and with IY as well as IX.

The BIT instructions are less interesting, undocumented codes DDCBjj4D though DDCBjj47 turn out to be equivalent to the documented version, no copying into registers B, C, D, etc. is done:

```
DD CB jj 40    BIT 0,(IX+jj)
DD CB jj 41    BIT 0,(IX+jj)
DD CB jj 42    BIT 0,(IX+jj)
DD CB jj 43    BIT 0,(IX+jj)
DD CB jj 44    BIT 0,(IX+jj)
DD CB jj 45    BIT 0,(IX+jj)
DD CB jj 46    BIT 0,(IX+jj) <documented>
DD CB jj 47    BIT 0,(IX+jj)
```

LOTS OF ROOM FOR MORE

My investigations into the ED group yielded little of interest. There were some duplications; for example, the eight instructions ED44, ED4C, ED54, ED5C, ED64, ED6C, ED74, ED7C all turned out to be NEG, even though only the first is documented as such. I also found duplicates for RETN, RETI, IM 0, IM 1, and IM 2. There were also a couple of "expected" duplicates:

```
ED 63 nn mm    LD (mmnn),HL
ED 6B nn mm    LD HL,(mmnn)
```

-- but these instructions already exist, and execute faster, in the non-ED set.

The ED group did provide a couple of lone curiosities:

```
ED 70          IN --, (C)
ED 71          OUT (C),--
```

These appear where you would expect the "missing" IN (HL),(C) and OUT (C),(HL). Nothing happens with (HL), though. IN --,(C) appears to function like IN A,(C), IN B,(C), etc., except that the result does not go anywhere. The flags, however, are set as expected. OUT (C),-- seems to output a zero to the port.

I could not detect any action for the first and fourth quarters of the ED set, ED00-ED3F and EDC0-EDFF. Three-fourths of the third quarter is also "missing", as are two instructions in the second quarter, ED77 and ED7F. This leaves room for 178 more instructions -- anyone for an upgrade? I'd like to see instructions like LD B,(DE) and CP (DE) and SUB (DE).

EXCEPTIONS

I said that any HL instruction could be changed to IX or IY by simply prefixing DD or FD. That was a little white lie. The following instructions are not convertible because they begin with ED:

```
ED 42          SBC HL,BC
ED 4A          ADC HL,BC
ED 52          SBC HL,DE
ED 5A          ADC HL,DE
ED 62          SBC HL,HL
ED 6A          ADC HL,HL
ED 72          SBC HL,SP
ED 7A          ADC HL,SP
ED 67          RRD <rotate right decimal (HL)>
ED 6F          RLD <rotate left decimal (HL)>
```

In addition, the following instructions cannot be converted even though they are one-byters:

```
D9             EXX <exchange all registers>
EB             EX DE,HL
```

-- and the JP (HL) instruction becomes JP (IX) or JP (IY), not JP (IX+jj) or JP (IY+jj):

```
   E9          JP (HL)
DD E9          JP (IX)
FD E9          JP (IY)
```

ONE AT A TIME, PLEASE

Except for DDCB and FDCB, there is no benefit in combining two or more of the four escape codes. There are three cases:

(1) In the event of multiple DDs and/or FDs, all but the last will be ignored.

(2) EDCB, EDDD, EDED, and EDFD will be ignored entirely since they lie in the inoperative fourth quarter of the ED set.

(3) If either DD or FD precedes ED, the former will be ignored.

A WORD OF CAUTION

To any machine-language programmers whose appetites may have been whetted: Most Z-80 Editor Assemblers do not recognize undocumented opcodes, so you'll have to enter these codes as DEFBs [since this article was written, some Editor Assemblers have appeared on the market that DO recognize the undocumented opcodes, although they may use different mnemonics than the ones given here -editor]. More important, it is conceivable that not all Z-80s will respond to these codes in the same way. If you plan to sell your programs, the best advice is not to use undocumented instructions [however, some TRS-80 disk operating systems now use undocumented opcodes to save memory space, and I have yet to hear even a rumor of a Z-80 that would not process these undocumented opcodes properly -editor].

SUMMARY OF USEFUL UNDOCUMENTED Z-80 OPCODES

DD24	INC HX	DD62	LD HX,D	DD8C	ADC A,HX
DD25	DEC HX	DD63	LD HX,E	DD8D	ADC A,LX
DD26nn	LD HX,nn	DD64	LD HX,HX	DD94	SUB HX
DD2C	INC LX	DD65	LD HX,LX	DD95	SUB LX
DD2D	DEC LX	DD67	LD HX,A	DD9C	SBC A,HX
DD2Enn	LD LX,nn	DD68	LD LX,B	DD9D	SBC A,LX
DD44	LD B,HX	DD69	LD LX,C	DDA4	AND HX
DD45	LD B,LX	DD6A	LD LX,D	DDA5	AND LX
DD4C	LD C,HX	DD6B	LD LX,E	DDAC	XOR HX
DD4D	LD C,LX	DD6C	LD LX,HX	DDAD	XOR LX
DD54	LD D,HX	DD6D	LD LX,LX	DDB4	OR HX
DD55	LD D,LX	DD6F	LD LX,A	DDB5	OR LX
DD5C	LD E,HX	DD7C	LD A,HX	DDBC	CP HX
DD5D	LD E,LX	DD7D	LD A,LX	DDBD	CP LX
DD60	LD HX,B	DD84	ADD A,HX		
DD61	LD HX,C	DD85	ADD A,LX		

The corresponding instructions for HY and LY may be obtained by using FD in place of DD.

CB30	SLL B	CB34	SLL H
CB31	SLL C	CB35	SLL L
CB32	SLL D	CB36	SLL (HL)
CB33	SLL E	CB37	SLL A

DDCBjj00-DDCBjj07	RLC r,(IX+jj)
DDCBjj08-DDCBjj0F	RRC r,(IX+jj)
DDCBjj10-DDCBjj17	RL r,(IX+jj)
DDCBjj18-DDCBjj1F	RR r,(IX+jj)
DDCBjj20-DDCBjj27	SLA r,(IX+jj)
DDCBjj28-DDCBjj2F	SRL r,(IX+jj)
DDCBjj30-DDCBjj37	SLL r,(IX+jj)
DDCBjj38-DDCBjj3F	SRL r,(IX+jj)
DDCBjj80-DDCBjj87	RES r,0,(IX+jj)
DDCBjj88-DDCBjj8F	RES r,1,(IX+jj)
DDCBjj90-DDCBjj97	RES r,2,(IX+jj)
DDCBjj98-DDCBjj9F	RES r,3,(IX+jj)
DDCBjjA0-DDCBjjA7	RES r,4,(IX+jj)
DDCBjjA8-DDCBjjAF	RES r,5,(IX+jj)
DDCBjjB0-DDCBjjB7	RES r,6,(IX+jj)
DDCBjjB8-DDCBjjBF	RES r,7,(IX+jj)
DDCBjjC0-DDCBjjC7	SET r,0,(IX+jj)
DDCBjjC8-DDCBjjCF	SET r,1,(IX+jj)
DDCBjjD0-DDCBjjD7	SET r,2,(IX+jj)
DDCBjjD8-DDCBjjDF	SET r,3,(IX+jj)
DDCBjjE0-DDCBjjE7	SET r,4,(IX+jj)
DDCBjjE8-DDCBjjEF	SET r,5,(IX+jj)
DDCBjjF0-DDCBjjF7	SET r,6,(IX+jj)
DDCBjjF8-DDCBjjFF	SET r,7,(IX+jj)

In the last 3 tables, the corresponding instructions for (IY+jj) may be obtained by using FD in place of DD. The value of r is

determined as follows:

Last digit of opcode:	register r:
0 or 8	B
1 or 9	C
2 or A	D
3 or B	E
4 or C	H
5 or D	L
6 or E	(blank)
7 or F	A

FOR/NEXT LOOP TERMINATION
by Ray Greet

[Reprinted from the Adelaide (South Australia) MICRO-USER News.]

About twelve months ago I mentioned to some members that exiting from FOR/NEXT loops prematurely, without terminating the loop, was a poor programming practice.

My justification of making the statement was based on information obtained from the IJG volume, MICROSOFT BASIC DECODED. This states that when a loop construct is generated, a 16 byte 'frame' is deposited on the system stack. From this I concluded that if a loop is not properly terminated then the construct frame would remain on the stack and so consume memory unnecessarily. The 'frame' is a structure containing details of loop parameters necessary for control of the loop process (this frame, incidentally expands to 20 bytes if a single precision index variable is used).

A recent listing passed to me which contains example of unterminated loops prompted me to research the matter further and report so as to clarify (hopefully) the situation as it would seem that the processes are not fully understood.

Some tests have shown that my original assumption was mostly correct; the frame remains on the stack except if a RETURN statement, when used as part of a subroutine, is the abortion device.

Any frame that is left on the stack will remain there until such time as a subsequent invocation occurs using the same reference variable. The significance of this is: if a subsequent allocation makes reference to a variable which was used by an earlier unterminated stack frame, that stack frame will be re-initialized. Any such re-initialization recovers all stack space that may be pending but used after any such abandoned frame. This means of course that any other pending FOR/NEXT frames will be lost, thus creating a source for NEXT without FOR errors. However, if a GOSUB is in control, the frame search function will only search the stack as far back as the GOSUB control frame, so pending loops invoked prior to a GOSUB are protected.

It seems obvious, that to avoid any possible confusion, good programming style should be adopted - terminate all loop constructs.

Some minor examples:

```
The WRONG way
10 FOR I%=1 TO 100
20  READ C%
30  IF C%=0 THEN 60
40  ...
50 NEXT

The RIGHT way (1)
10 FOR I%=1 TO 100
20  READ C%
30  IF C%=0 THEN I%=100:GOTO 50
40  ...
50 NEXT
60 ...

The RIGHT way (2)
10 FOR I%=1 TO 100
20  READ C%
30  IF C%=0 THEN I%=100:NEXT:GOTO 100
40  ...
50 NEXT
60 ...
100

The RIGHT way (3)
10 GOSUB 30
20 STOP
30 FOR I%=1 TO 100
40  READ C%
50  IF C%=0 THEN RETURN
60  ...
70 NEXT
80 RETURN
```

The first RIGHT way example would be used when code continuation is required, the second if further branching is necessary. The abortive method is used in example three is quite acceptable as the RETURN statement will automatically purge the abandoned FOR/NEXT frame from the stack. The method by which the interpreter can clobber loop conditions as described earlier is a design flaw. It would be more logical to flag as an error any attempt to open a loop with a variable already assigned to a loop. Whether this has been changed in later versions is unknown to me.

HOW TO WIN AT ANDROID NIM
by Murray Hayman

[Remember Android Nim? It was one of the very first games for the TRS-80 to make use of animation-type graphics. Although it would be considered crude by today's standards, at the time a lot of folks were amazed that you could do such graphics-intensive program on a "lowly" TRS-80. The game itself wasn't all that easy to win, either. For those of us that never were successful, here's the solution (after all these years!). This article was reprinted from the APCU (Association of Personal Computer Users) Newsletter.]

There are many different ways of playing Nim. This concerns Android NIM only, which has a complete mathematical analysis. Any number of Androids are divided arbitrarily among three rows. Two players, or the computer and one, play alternately. Each in his turn may select any row, and remove from it as many (at least one) Androids as desired, or all of them in the row. The player who takes the last Android wins.

In any NIM there are what are called safe or unsafe combinations. A safe combination for you occurs after you have removed Androids from a row you have left the remaining positions in a safe combination and force the opponent to make it unsafe for himself by any play.

The game starts with an unsafe combination with 7 Androids in the 1st row, 5 in the 2nd row, 3 in the third. It is determined to be unsafe by expressing the number of Androids in each row in binary form then adding the corresponding bits without carrying and getting a '1' in the sum. We seek a combination that adds up to all zeroes.

start	possible plays	
7=111	6=110	7=111
5=101	5=101	4=100
3=011	3=011	3=011
-----	-----	-----
001	000	000
unsafe	safe	safe

You can see from the above that you should go first if you have that option. Then you can remove one Android from any of the rows so the binary digits in each column will add to zero.

Now no matter what the opponent does it will make it unsafe for itself, and you can then see which move will produce a safe combination again.

There is a way to perform the no-carry binary addition using your fingers. I use my right hand palm down and designate my little finger as '1' binary, ring finger as '2', and little finger as '4'. The bottom row at start has 3 Androids, or 011, so I'd raise the little finger and ring finger. The middle row has 5, or 101, so I'd raise the middle finger and lower the raised pinkie. Now the middle and ring fingers are raised, representing 110, or 6. Now to get a safe combination, I would need to add a 6, or 110, to the 110 I have in the raised fingers. This is done by removing one Android from the seven in the top row. Adding the remaining 6 makes it a safe combination, represented by lowering the two raised fingers in the addition. All fihgers down is the safe combination.

The opponent can make no safe move from my safe combination, so will make it unsafe again. By the same approach I can always make it safe again, until there remain only 1, 2 or 3 Androids in a single row which I can remove to win. Incidentally, a safe combination exists if there are only two equal rows.

In this version, the safe combinations are:

7	7	6	5	5	4	3	3	2	1
6	4	5	4	5	4	3	2	2	1
1	3	3	1				1		

The Alternate Source and Northern Bytes PACKAGE DEAL

TRS-80 Software Support for New Users

One complete edition, Volume 5, of Northern Bytes.

The Northern Bytes newsletter provides the perfect medium for exchanging technical information for the world's most popular computers! This offer includes the latest EIGHT issues, the complete printing for 1984.

PLUS

The Alternate Source Programmer's Journal Special Edition #19

including a two-sided "flippy" disk (that may be used with any single sided 40-track drive) with many usuable routines and programs that you can use in your own programs!

PLUS

Four flippy diskettes that are loaded with public domain software that you can share with your friends. These diskettes comprise the first four volumes of The Alternate Source TRS-80 Public Domain Library.

This entire package would cost $75.95 if purchased separately. Order the package deal for only $59.95 and SAVE $16 dollars.

Mail To: THE ALTERNATE SOURCE, 704 N. Pennsylvania, Lansing, MI. 48906
Phone (517) 482-8270 for VISA, MasterCard and COD orders.

Your name: _____
Address: _____
Address: _____
Address: _____
Country: _____

Please add $3 to your entire order for shipping and handling.
Michigan residents please add 4% sales tax

- ☐ Complete Package Deal ($59.95)
- ☐ Public Domain Volumes ($10 each)
  Please indicate which volumes (1-4):
- ☐ The Alternate Source Programmer's Journal #19 ($19.95)
- ☐ Northern Bytes Volume 5 Complete
- ☐ Issues of Northern Bytes ($2 each)
  Please indicate which issues:
- ☐ Please leave my name off list rentals

I enclose $ _____
☐ VISA ☐ MasterCard Account # _____
Expiration date and Signature _____
All orders must be paid in U.S. funds drawn on a U.S. bank.

$49.95 For TRS-80 Models I, III and 4(III)

Text Press

A Text Archival System

- Quickly Compress and Decompress
- Define Your Own Word List
- View Files Without Decompression
- Process one or several files at once
- File Overwrite Protection
- Detailed File Statistics
- Excellent User Manual

Distributed by:
The Alternate Source
704 North Pennsylvania Avenue
Lansing, MI 48906
(517) 482-8270

NORTHERN BYTES
c/o Jack Decker
1804 West 18th Street
Lot # 155
Sault Ste. Marie, Michigan 49783
MCI Mail Address: 102-7413
Telex: 6501027413
(Answerback: 6501027413 MCI)

Bulk Mail
U.S. Postage Paid
Permit #815
Lansing, MI

POSTMASTER: If undeliverable return to:
The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906

To: